The Persistent Socket Datagram Protocol: Concepts, Security and Reliability (DRAFT)

Eike Scholz <es@polygravity.io>

September 14, 2018

> In order to be an immaculate member of a flock of sheep, one must above all be a sheep oneself.
>
> *(Albert Einstein)*

This paper introduces the Persistent Socket Datagram Protocol (PSDP) and provides a security and reliability assessment, all with formal methods. The PSPD provides an encrypted, authenticated, reliable, DDoS resilient, real-time, low latency, massive throughput, byzantine failure tolerant, message oriented transmission layer protocol based on IPv6. The protocol defines message exchange between byzantine failure tolerant, high availability clusters created by replication and consisting of at least a single node. While the given formal assessment is conservative it already demonstrates the ability to satisfy military grade security and reliability requirements. The document concludes with the definition of send and receive algorithms that allow massive throughput and low latency traffic handling and demonstrate the excellent capacity of the protocol to build cluster local blockchain applications.

Contents

1. Introduction

1.1. Overview

Arguably the simplest solution to provide byzantine fault tolerant services is replication [1]. The Persistent socket datagram protocol provides a solution to the problem of transferring the byzantine fault tolerance of two replicated, low latency, massive throughput services to a reliable, encrypted and authenticated communication protocol between them. At least, due to their limited 16-Bit port range, the commonly used UDP [3] and TCP [2] Internet transmission layer protocols lack in massive throughput capabilities. This fact and required suitability for formal verification to meet military grade requirements where the main reasons for the development of the persistent socket datagram protocol as a new transmission layer protocol based on IPv6[1] [4]. While not required for many applications the real-time requirement in this protocol is not disposable since is necessary to define the communication logic[2]. If real-time limits are set, latency becomes an issue. Thus to save cryptographic handshake overhead time connections are made persistent. The structure this document is:

1. Provide this introduction.
2. Give a formal model for the here required parts of the Internet.
3. Give a formal model an Interface to the cryptographic algorithms.
4. Give a formal specification of PSDP Packets and their Semantics
5. Define the Persistent Socket Datagram Protocol.
6. Give a Conservative security and reliability assessment with formal methods.
7. Define parallel, low-latency, massive-throughput, IO handling algorithms.
8. A mathematical appendix of theory developed to perform the formal assessments.

1.2. Formal Notations

Regarding the formal mathematical nomenclature proofs are terminated with a box symbols and the following naming scheme is used:

Lemma: A useful statement of insight into formal specifics.

Theorem: A statement of universal insight or usefulness.

Corollary: A statement trivial to prove. The proof is omitted if the required results to prove it have been, in reading order terms, recently stated.

[1] Using IPv4 [5] is discouraged in this case. The IPv4 fragmentation handling is complex and not well suited for formal verification. If required, a minimal verified IPv6 interface is much cheaper to produce.

[2] Meaning its integral and not removable, and thus a restriction, but it sounds like a feature.

Meta Definition: A definition expressed in non formalized clear language. It is used for describing general theories that are applicable to any formal instantiation of the meta definitions. That is, to apply a general theory expressed with meta definitions have to be formally instantiated to the application context. Usually meta definitions are omitted by mathematician[3] and rigor definitions stating formally necessary formally expressible, often purely algebraic properties, which after several instances of abstraction conceals how to apply a theory in a context. Meta definitions are the link from an analytic philosophical clear plain language definitions to their rigor formalizations.

This document aims too provide all formal or mathematical definitions and results required to check the proofs, aside from basic formal set theory and school mathematics. In that regard anyone with training in basic formal methods should be able to check the proofs. However the modeling style is motivated by formal statistics and stochastics and familiarity especially with mathematical measure theory will help. Furthermore the following not so common notations are used

$\#M$ **:** The principal counting measure[4] of M, that is number of elements if M is finite countable.

$a{+}{+}b$ **:** The concatenation of two tuples. For example $(a,b,c){+}{+}(x,y) = (a,b,c,x,y)$.

$\lfloor X \rceil$ **:** X rounded down to the nearest integer.

$\bot$ **:** Error indicator value, usually denotes absurdity in formal logic.

$\mapsto$: Reads "maps to" and is an anonymous function definition. E.g. the function that adds the outer elements of a triple can be written $(a,b,c) \mapsto a + c$ without giving it a name. The document already names a lot of functions.

Nat_z **:** A z-Bit big-endian representation of natural numbers smaller then 2^z.

Int_z **:** A z-Bit big-endian representation of integers bigger then -2^{z-1} smaller then 2^{z-1}.

$f(X)$ **:** The usual Function application if X is in the domain of f. Otherwise if X is a subset of the domain of X the point-wise function application, that is the set $\{f(x) | X \in X\}$.

$[y]^f_A$ **:** Preimage of a function with restricted domain, formally defined as

$$[y]^f_A := \{x \in A | f(x) = y\} \ . \tag{1}$$

The notation is intentionally similar to the common equivalence class notation, only that y is not a represent but a identifier. The notation denotes equivalence cases for the relation $p \sim_f q :\Leftrightarrow f(p) = f(q)$

[3] To be fair, in applied mathematics they are often and in pure mathematics sometimes explained in the "fluff-text" of mathematical treatises.

[4] Commonly just called the counting measure. See the appendix on meassure theory for an explanation of the qualification "principal".

$A/_f$ **:** The set of equivalence classes created with f, formally defined as

$$A/_f := \{[y]_A^f | y \in f(A)\} \ . \tag{2}$$

This is an abbreviation for the common notation $A/\sim_f$ using the relation $\sim_f$ defined above.

2^X **:** The usual meaning if X is a number and the power set if X is a set.

$\underline{N}$ **:** The index set defined as $\underline{N} := \{0, 1, 2, \ldots, N-1\}$ for $N \in \mathbb{N}$.

$\oplus$ **:** The exclusive or operation, performed bitwise where applicable.

$\bigcup F$ **:** Union of subsets i.e. short for the common notation

$$\bigcup_{I \in F} I$$

, also formally the letter notation can be defined as an abbreviation of the former combined with the set comprehension syntax. So if $F \in 2^{2^A}$, i.e. $F \subseteq 2^A$ then $\bigcup F \in A$.

Other notations, and some of these for that matter, should be introduced in basic algebra and analysis courses of any STEM and economics undergrad education[5].

[5]This is still a draft and a work in progress, don't mind to ask the author, some things are probably missing here, and non of your educators is to blame.

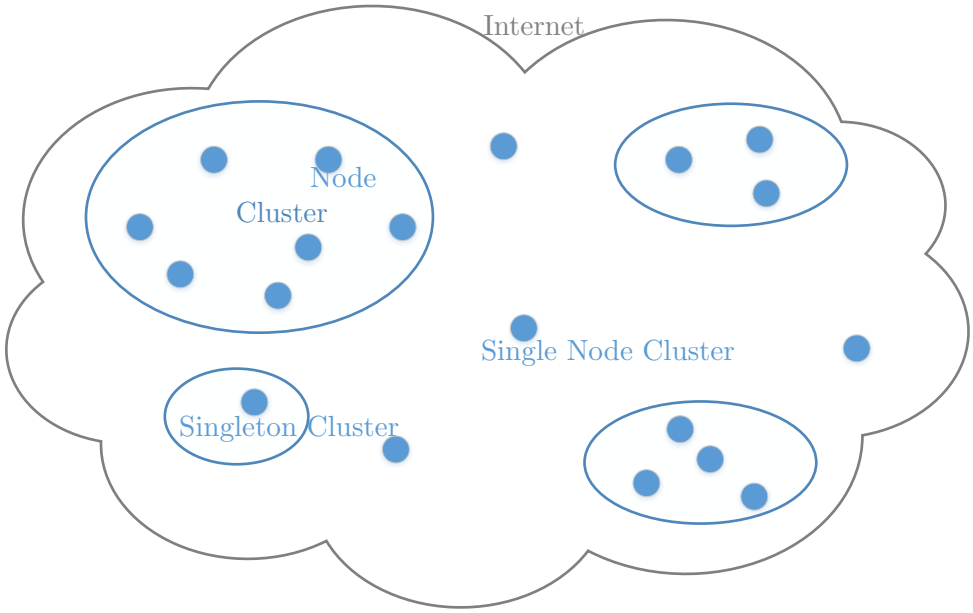

Figure 1: The Internet Participants

2. The Internet from the PSDP Perspective

With respect to the PSPD the Internet looks like figure 1. For the purpose of this document the Internet protocol is the Internet Protocol version 6 (IPv6). The here important components of the Internet are given by:

2.1. Addresses

Definition 2.1.1 (Internet Address)**.** *A number in* Nat_{128}*.*

Definition 2.1.2 (Unicast Address)**.** *An Internet address that refers to a specific ipv6 interface. The set of all unicast addresses is denoted* $\mathcal{N}$

Definition 2.1.3 (Multicast Address)**.** *A Internet address that refers to a set of unicast addresses.*

2.2. Nodes

Definition 2.2.1 (Node)**.** *A networked computer with dedicated IPv6 unicast address.*

2.3. Clusters

Clusters form the PSDP perspective run replicated high availability, byzantine failure tolerant services, so that data received though the protocol ought be the same on all notes of a cluster. Other kinds of cluster may exist but are of no relevance for the PSDP.

Definition 2.3.1 (Cluster:)**.** *A family, i.e. indexed set or variable length tuple, of nodes with dedicated IPv6 address, that must be a multicast address if the cluster consists of more then one node. If the cluster consists of only one node and adding further nodes is not required, the cluster can use the nodes dedicated non-multicast address as address. The nodes in a cluster will indexed from 0. For example, a cluster A with 3 nodes, consists of node 0, node 1 and node 2. The set of all possible clusters is defined as*

$$\mathcal{C} := \bigcup_{n=1}^{2^{128}} \mathrm{Nat}_{128}^{n}. \tag{3}$$

where the biggest clusters in this set are technically impossible. However the empty cluster is not included. All clusters $c \in C$ ought be interpretable as sets, so that here are no duplicate elements, that means that

$$\forall c \in \mathcal{C} : c_i = c_j \Rightarrow i = j \tag{4}$$

In most cases the elements of C a simply treaded as sets, even if they formally are not, but adding a function that maps for from a cluster to the set of its members is trivial formal clutter otherwise.

Definition 2.3.2 (Singleton Cluster:)**.** *A cluster consisting of a single node with an dedicated multi-cast address. A singleton cluster can be extended with additional nodes.*

Definition 2.3.3 (Single Node Cluster:)**.** *A cluster consisting of a single node without dedicated multi-cast address. A singleton cluster can not be be extended with additional nodes.*

Definition 2.3.4 (Node Id)**.** *Since cluster have different nodes, the following node id function is well defined*

$$\begin{aligned} &\mathrm{nodeId} : \mathcal{N} \to \mathbb{N} \\ &\mathrm{nodeId} := n \mapsto \Big\{ i \text{ if } n = \mathrm{clusterOf}(n)_i \end{aligned} \tag{5}$$

By construction of clusterOf *the condition can allways be satisfied so there is no otherwise-clause in the above function case decomposition.*

2.4. Packets

Definition 2.4.1 (Packet)**.** *The actual IPv6 packets send between nodes. It essentially consists of the source and destination address for the data transported. The additional data is relevant technical specifics of data transportation and processing.*

2.5. Sockets

The IPv6 only specifies the interfaces a packets comes from and goes to using addresses. It does not specify the program the information was sent from or is send to. Specifying this is left to higher layer protocols like UDP,TCP or PSDP, however it is common to solve this problem using the following concepts:

Definition 2.5.1 (Socket and Port)**.** *A socket is a communication endpoint and is identified by a pair consisting of one Internet address and one number, called Port, identifying the actual endpoint in a program. For example, the PSDP uses* Nat_{64} *numbers for port numbers, while the common UDP and TCP use* Nat_{16} *for port numbers.*

2.6. Setup

Definition 2.6.1 (Valid Cluster Setup)**.** *Getting the cluster of an address is done with the following function*

$$\mathrm{clusterOf} : \mathrm{Nat}_{128} \to \mathcal{C} \ . \tag{6}$$

Each node is at least member of a cluster containing the node itself. That is, the condition

$$\forall n \in \mathcal{N} : n \in \mathrm{clusterOf}(n) \ . \tag{7}$$

must hold. However the clusters multicast addresses must not be included, that is

$$\forall n \in \mathrm{Nat}_{128} \backslash \mathcal{N} : n \notin \mathrm{clusterOf}(n) \ . \tag{8}$$

Also we demand for a valid cluster setup that a node is member of at most one cluster, that is the following holds

$$\forall c, c' \in \mathcal{N} : c \cap c' = \varnothing \ . \tag{9}$$

Definition 2.6.2 (Address Assignment)**.** *Getting the multi-cast address of a cluster is done with*

$$\mathrm{addrOf} : \mathcal{C} \to \mathrm{Nat}_{128} \tag{10}$$

A valid address assignment must satisfy the following conditions

$$\forall n \in \mathrm{Nat}_{128} : \mathrm{addrOf}(\mathrm{clusterOf}(n)) = n \tag{11}$$

For the above the following abbreviation is used later:

Definition 2.6.3. *Getting the multi-cast address of a cluster is done with*

$$\mathrm{clusterAddrOf} : \mathcal{N} \to C \tag{12}$$

$$\mathrm{clusterAddrOf} = n \mapsto \mathrm{addrOf}(\mathrm{clusterOf}(n)) \tag{13}$$

3. Cryptography from the PSDP Perspective

The PSDP does not specify any specific cryptographic algorithms, instead it uses cryptography though a general abstract interface. The below functions have as a subscript the type of data processed. For example enc_A is the function for encryption of values in A. This subscript is omitted if it is inferable from the argument, so that the below definitions are a polymorphic functional programming interface[6] expressed in common mathematical notation.

3.1. Keys

Definition 3.1.1 (The set of all Cryptographic Keys)**.** *The set of all cryptographic keys is denoted $\mathcal{K}$. Any element in K is not only the key of an algorithm, it also contains an ID of the specific algorithms it belongs to.*

3.2. Encryption

Definition 3.2.1 (Encryption)**.** *The encryption interface is given by*

$$\text{enc}_A : A \times \text{Nat}_{192} \times \mathcal{K} \times \mathcal{K} \to A \cup \{\bot\} \tag{14}$$

where $\bot$ is returned if the keys provided are not a (sender,receiver) key pair of an encryption algorithm. The first argument is the text to encrypt, the second is a number used once (nonce) big enough to be generated randomly with negligible collision probability[7], the first key is the senders key the second key is the receivers key.

$$\text{dec}_A : A \times \text{Nat}_{192} \times \mathcal{K} \times \mathcal{K} \to A \cup \{\bot\} \tag{15}$$

where $\bot$ is returned if the keys provided are not a sender receiver key pair of an encryption algorithm.

3.3. Digital Signatures

Definition 3.3.1 (Digital Signatures)**.** *The digital signatures are generated with[8]*

$$\text{mksig}_A : A \times \mathcal{K} \to \text{Nat}_{512} \cup \{\bot\} \tag{16}$$

where $\bot$ is returned if the key provied is not the secret key for a signature algorithm. Signatures are checked for validity with

$$\text{cksig}_A : A \times \mathcal{K} \times \text{Nat}_{512} \times \to \{0, 1, \bot\} \tag{17}$$

[6] This interface needs some improvements to be used as a software interface, but is sufficient for the formal analysis at hand.

[7] This is claim found in the libsodium online documentation at least until Sep. 2018. The author was not able to find a Paper by a professional cryptographer to support this claim. However the birthday theorem roughly yields that the collision probability for 2^{64} created 192-bit nonces is about 2^{-64}.

[8] Are there useful signature algorithms that use nonces? Then a nonce parameter should be included.

where 0 *is returned if the signature check fails* 1 *if it succeeds and* $\perp$ *if the key provided is not a public key of a signature algorithm. Algorithms that need to output more then 512 significant bits are considered broken by design*[9] *and not included.*

3.4. Authentication Codes

Definition 3.4.1 (Authentication Codes)**.** *Authentication codes*[10] *are similar to signatures with the difference that they use a shared secret key and a nonce. They should fast and have small footprint*

$$\text{mkac}_A : A \times \text{Nat}_{128} \times \mathcal{K} \to \text{Nat}_{128} \cup \{\perp\} \tag{18}$$

where $\perp$ *is returned if the key provied is not the public key for a signature algorithms. Authentication codes use a single shared key so that a dedicated check function is not necessary, and checking for authentication is comparing authentication codes.*

3.5. Key Infrastructre

The PSDP is agnostic about any details of the used cryptographic algorithms. However the setup of the nodes and clusters needs to provide the locally needed information of the functions of a valid cryptographic setup. However the specifics of how this is done are not of interest in this paper, instead the information is accessed using the following functions:

Definition 3.5.1 (Valid Cryptographic Setup)**.** *There implicitly exist abstract functions so that*

$$\text{sockSigPK} : \mathcal{N} \times \text{Nat}_{64} \to \mathcal{K} \tag{19}$$

returns public signature key for a socket,

$$\text{sockSigSK} : \mathcal{N} \times \text{Nat}_{64} \to \mathcal{K} \tag{20}$$

returns secret signature key for a socket,

$$\text{sockEncPK} : \text{Nat}_{128} \times \text{Nat}_{64} \to \mathcal{K} \tag{21}$$

returns public encryption key for a socket,

$$\text{sockEncSK} : \text{Nat}_{128} \times \text{Nat}_{64} \to \mathcal{K} \tag{22}$$

returns secret encryption key for a socket,

$$\text{nodeEncPK} : \text{Nat}_{128} \to \mathcal{K} \tag{23}$$

[9] Just compare 2^{512} to the number of possible quantum states in the visible universe, to see why.

[10] Usually referred to as message authentication codes (mac), but they are not used here to authenticate messages.

returns public encryption key of a node,

$$\text{nodeEncSK} : \text{Nat}_{128} \to \mathcal{K} \tag{24}$$

returns secret encryption key of a node,

$$\text{nodeAK} : \mathcal{N} \to \mathcal{K} \tag{25}$$

returns the node authentication key used to construct route authentication codes. All keys should be mutually different. Since replication implies that a single broken encryption key allows to read the data in the replication network the following requirements key sharing requirements are added to simplify multicast decryption without relevant[11] security loss:

$$\forall C \in \mathcal{C} : \forall n, n' \in C : \text{nodeEncPK}(n) = \text{nodeEncPK}(n') \tag{26}$$

$$\forall C \in \mathcal{C} : \forall n, n' \in C : \text{nodeEncSK}(n) = \text{nodeEncSK}(n') \tag{27}$$

$$\forall C \in \mathcal{C} : \forall n, n' \in C : \forall p \in \text{Nat}_{64} : \text{sockEncPK}(n, p) = \text{sockEncPK}(n', p) \tag{28}$$

$$\forall C \in \mathcal{C} : \forall n, n' \in C : \forall p \in \text{Nat}_{64} : \text{sockEncSK}(n, p) = \text{sockEncSK}(n', p) \tag{29}$$

Furthermore it is required that the key data of nodeAK *has an even size, so that it can be used to generate route authentication keys as described below. Last it is required the associated algorithms will not fail with $\perp$ on the corresponding keys[12].*

This key infrastructure is defined to define Connections in the PSDP by having finished a proper key exchange. After this the keys ought to be stored persistently on the nodes that require them to ensure a low latency through the ability to omit any key exchange steps in the protocol. This makes the connections and the associated connection endpoints, called sockets, persistent as well. A file system representation of the persistent sockets and their required associated cryptographic data similar to the sys and proc file system of linux, but implementable as plain file system entry, will be specified in an other document.

3.6. Route Authentication Codes

Definition 3.6.1 (Route Authentication Key)**.** *Node authentication keys have the special property that they can be split into equally sized pairs. Then for to nodes n, n' there exist $(l, r) = \text{nodeAK}(n)$ and $(l', r') = \text{nodeAK}(n')$ and the route authentication key for the route form n to n' is then defined as $\text{routeAK}(n, n') := (l \oplus r', r \oplus l')$ where $\oplus$ is the xor operation.*

Definition 3.6.2 (Route Authentication Code (rac))**.** *The authentication codes created with route authentication keys are route authentication codes (rac).*

[11] If that a loss at all, since higher algorithm complexities increase potential attack surface due to higher probability to write buggy implementation.

[12] That is to keep tedious error handling out of an specification. An implementation usably can not guarantee that, since at the least memory resources might be exhausted and cause an error condition.

4. The PSDP Packet and its Semantics

4.1. Packet Structure

Definition 4.1.1 (PSDP Packet)**.** *A PSDP packet is any element of the set*

$$\mathcal{P}_{\text{psd}} := \mathcal{H}_6 \times \mathcal{H}_\text{n} \times \mathcal{H}_\text{s} \times \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \times \mathcal{F}_\text{i} \times \mathcal{F}_{\text{rac}} \tag{30}$$

where $\mathcal{H}_6$ is the set of IPv6 headers, $\mathcal{H}_\text{n}$ is set of node encryption segment headers, $\mathcal{H}_\text{s}$ is the set of socket encryption segment headers, $\mathcal{H}_p$ is the set of payload prefixes, $\mathcal{P}$ is the set of payloads, $\mathcal{F}_{\text{sig}}$ is the set of signature footers,$\mathcal{F}_\text{i}$ is the set of initial source addresses ,$\mathcal{F}_{\text{rac}}$ is the set of route authentication code footers and these sets are defined by:

$$\mathcal{H}_6 := \text{Nat}_4 \times \text{Nat}_8 \times \text{Nat}_{20} \times \text{Nat}_{16} \times \text{Nat}_{128} \times \text{Nat}_{128} \tag{31}$$

$$\mathcal{H}_\text{n} := \text{Nat}_{192} \tag{32}$$

$$\mathcal{H}_\text{s} := \text{Nat}_{64} \times \text{Nat}_{64} \tag{33}$$

$$\mathcal{H}_\text{p} := \text{Int}_{64} \tag{34}$$

$$\mathcal{P} := \bigcup_{n=0}^{1096} \text{Nat}_8^n \tag{35}$$

$$\mathcal{F}_{\text{sig}} := \text{Nat}_{512} \tag{36}$$

$$\mathcal{F}_\text{i} := \mathcal{N} \tag{37}$$

$$\mathcal{F}_{\text{rac}} := \text{Nat}_{128} \tag{38}$$

Due to the size limit of the payload a packet is never greater then the guaranteed MPU for IPv6. Applications and protocols build on top never need to do mtu path discovery[13].

4.2. Header Semantics

4.2.1. IPv6 Header Semantics

Let $(v, t, f, l, n, h, s, d) \in \mathcal{H}_6$ then $v = 6$ is the protocol version, t is the traffic class, f is the flow label, l is the ipv6 packet payload length, n is the next header type id, s is the source address and d is the destination address. More detailed semantic can be found in the IPv6 Specification [4]. For the persistent socket datagram protocol n = PSD_PROTO_NUM which is currently 59 (No Next Header) as long as the PSDP has no officially registered protocol number.

$$\begin{aligned} &\text{hdr}_6 : \mathcal{P}_{\text{psd}} \to \mathcal{H}_6 \\ &\text{hdr}_6 := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto h_6 \end{aligned} \tag{39}$$

[13]This restriction may be removed or relaxed later. It is technically unnecessary. However it allows very simple efficient implementations. With this restriction implemented packet buffers do not, in practice, need dynamic memory management. So this restriction is there to specify attack surface of implementations away. I would like to have some feedback from experienced network engineers if this is a good design decision.

$$\begin{aligned}&\text{srcAddr} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{128}\\&\text{srcAddr} := ((v,t,f,l,n,h,s,d),h_n,h_s,h_p,m,f_s,f_i,f_r) \mapsto s\end{aligned} \tag{40}$$

$$\begin{aligned}&\text{dstAddr} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{128}\\&\text{dstAddr} := ((v,t,f,l,n,h,s,d),h_n,h_s,h_p,m,f_s,f_i,f_r) \mapsto d\end{aligned} \tag{41}$$

$$\begin{aligned}&\text{length} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{16}\\&\text{length} := ((v,t,f,l,n,h,s,d),h_n,h_s,h_p,m,f_s,f_i,f_r) \mapsto l\end{aligned} \tag{42}$$

4.2.2. Node Encryption Segment Header Semantics

Let $n \in \mathcal{H}_{\text{n}}$ then n is the seed nonce of the datagram, a number used to generate actual nonces used once per node encryption key. It can be generated randomly or contain an encrypted or hashed message id and message part information.

$$\begin{aligned}&\text{hdr}_{\text{n}} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_6\\&\text{hdr}_{\text{n}} := (h_6,h_n,h_s,h_p,m,f_s,f_i,f_r) \mapsto h_n\end{aligned} \tag{43}$$

The nonces used actually used for node and socket encryption are generated[14] using data from the next headers. Specifically for node segment encryption the nonce is generated by

$$\begin{aligned}&\text{neNonce} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{192}\\&\text{neNonce} := ((v,t,f,l,n,h,s,d),h_n,(s_p,d_p),h_p,m,f_s,f_i,f_r)\\&\qquad\qquad \mapsto h_n \oplus ((\text{idst}(s,d) \oplus (s_p{+\!+}s_p)){+\!+}d_p)\end{aligned} \tag{44}$$

where the helper function "initial destination"

$$\begin{aligned}&\text{idst} : \text{Nat}_{128} \times \text{Nat}_{128} \to \text{Nat}_{128}\\&\text{idst} := (s,d) \mapsto \begin{cases} d & \text{if } d \in \mathcal{N} \\ s & \text{otherwise} \end{cases}\end{aligned} \tag{45}$$

whose name will be explained by later definitions is used. For socket segment encryption the nonce is generated by

[14] This is a placeholder generation with appropriate properties as existence proof, but will likely replaced with something aiding distribution of the same seed nonce to all nodes of a cluster. Maybe some concatenation of sender local nonces to form the seed nonce and then some clever rol- and ror-ing to generate the node and socket nonces.

$$\begin{aligned}&\text{seNonce} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_6\\&\text{seNonce} := ((v,t,f,l,n,h,s,d),h_n,(s_p,d_p),h_p,m,f_s,f_i,f_r) \mapsto h_n \oplus (f_i{+\!+}s_p)\end{aligned} \tag{46}$$

As the name suggest a of the packet segment after this header will be encrypted using node specific algorithms and keys.

4.2.3. Socket Encryption Segment Header Semantics

Let $(s_p, d_p) \in \mathcal{H}_\text{s}$ then s_p is the source port and d_p is destination port. This header implies the following interface:

$$\begin{aligned}&\text{hdr}_\text{s} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_6\\&\text{hdr}_\text{s} := (h_6,h_n,h_s,h_p,m,f_s,f_i,f_r) \mapsto h_s\end{aligned} \tag{47}$$

$$\begin{aligned}&\text{srcPort} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{64}\\&\text{srcPort} := (h_6,h_n,(s_p,d_p),h_p,m,f_s,f_i,f_r) \mapsto s_p\end{aligned} \tag{48}$$

$$\begin{aligned}&\text{dstPort} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{64}\\&\text{dstPort} := (h_6,h_n,(s_p,d_p),h_p,m,f_s,f_i,f_r) \mapsto d_p\end{aligned} \tag{49}$$

$$\begin{aligned}&\text{src} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{64} \times \text{Nat}_{64}\\&\text{src} := ((v,t,f,l,n,h,s,d),h_n,(s_p,d_p),h_p,m,f_s,f_i,f_r) \mapsto (s,s_p)\end{aligned} \tag{50}$$

$$\begin{aligned}&\text{dst} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{64} \times \text{Nat}_{64}\\&\text{dst} := ((v,t,f,l,n,h,s,d),h_n,(s_p,d_p),h_p,m,f_s,f_i,f_r) \mapsto (d,d_p)\end{aligned} \tag{51}$$

As the name suggest a segment of the packet after this header will be encrypted using socket specific algorithms and keys.

4.2.4. Payload Prefix Semantics

Let $t \in \mathcal{H}_\text{c}$ then t is a deadline measured in microseconds since the unix epoch (UTC), It is encrypted together with the payload with socket specific source node encryption keys.

$$\begin{aligned}&\text{hdr}_\text{p} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_6\\&\text{hdr}_\text{p} := (h_6,h_n,h_s,h_p,m,f_s,f_i,f_r) \mapsto h_p\end{aligned} \tag{52}$$

A deadline is considered passed and the packet is dropped if the deadline has passed after the decryption of a packet. So arriving at the interface in time does not guarantee that the deadline will be met. Systems have to be constructed with deadlines keeping this in mind. Also on systems with insufficient decryption resources, matching a deadline may be dependent on the system load[15].

[15]This simplifies implementation, but may be dropped if this design decision is shown to be problematic.

4.2.5. Signature Footer Semantics

Let $s \in \mathcal{F}_{\text{sig}}$ then s is a digital signature generated with a socket specific secret key and signature algorithm.

$$\begin{aligned} &\text{ftr}_\text{s} : \mathcal{P}_{\text{psd}} \to \mathcal{F}_{\text{sig}} \\ &\text{ftr}_\text{s} := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto f_s \end{aligned} \tag{53}$$

4.2.6. Initial Source Address Footer Semantics

Let $a \in \mathcal{F}_\text{i}$ then a is the address used to create the signature in the signature footer, since the source header field changes in a later stage of the PSDP protocol as described below.

$$\begin{aligned} &\text{sgnrAddr} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{128} \\ &\text{sgnrAddr} := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto f_i \end{aligned} \tag{54}$$

$$\begin{aligned} &\text{sgnr} : \mathcal{P}_{\text{psd}} \to \text{Nat}_{128} \times \text{Nat}_{512} \\ &\text{sgnr} := (h_6, h_n, (s_p, d_p), h_p, m, s, f_i, f_r) \mapsto (f_i, s_p) \end{aligned} \tag{55}$$

4.2.7. Route Authentication Code Semantics

This footer provides a dedicated distributed denial of service mitigation mechanism. It is a single value $c \in \mathcal{F}_{\text{rac}}$.

$$\begin{aligned} &\text{ftr}_\text{r} : \mathcal{P}_{\text{psd}} \to \mathcal{F}_{\text{sig}} \\ &\text{ftr}_\text{r} := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto f_r \end{aligned} \tag{56}$$

4.3. The Packet Normal Form

The packet normal form defaults some IPv6 header fields, that may change during transmission. It further "corrects" the payload length field if wrong.

Definition 4.3.1. *The packet normal form of a packet is given by*

$$\begin{aligned} &\text{nf} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{nf} := ((v, t, f, l, n, h, s, d), h_n, h_s, h_p, m, f_s, f_r) \\ &\qquad \mapsto (\,(6, 0, 0, \\ &\qquad\qquad \mathit{length}(m) + N_{\text{ip6}}^{\text{hdr}} + N_{\text{sig}}^{\text{ftr}} + N_{\text{i}}^{\text{ftr}} + N_{\text{rac}}^{\text{ftr}}, \\ &\qquad\qquad s, d\,), h_n, h_s, h_p, m, f_s, f_r) \end{aligned} \tag{57}$$

where $N_{\text{ip6}}^{\text{hdr}}$ is the IPv6 header size, $N_{\text{sig}}^{\text{ftr}}$ the signature footer size, $N_{\text{i}}^{\text{ftr}}$ the initial source footer size and $N_{\text{rac}}^{\text{ftr}}$ the rac footer size.

Corollary 4.3.1 (The Normal Form is Idempotent)**.**

$$\forall p \in \mathcal{P}_{\text{psd}} : \text{nf}(\text{nf}(p)) = \text{nf}(p) \tag{58}$$

4.4. Encrypted Packets

The complete packet encryption is a two layered encryption given by the functions

$$\begin{aligned} &\text{pktEnc} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{pktEnc} := p \mapsto \text{nodeEnc}(\text{sockEnc}(\text{nf}(p))) \end{aligned} \tag{59}$$

$$\begin{aligned} &\text{pktDec} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{pktDec} := p \mapsto \text{sockDec}(\text{nodeDec}(\text{nf}(p))) \end{aligned} \tag{60}$$

where the contained socket and node specific functions are explained next. While probably not accidentally constructible sockEnc should not correspond to some kind of right inverse of nodeEnc. With regard to that, do not trust proprietary non-disclosed encryption algorithms. It is not that hard to construct pairs of algorithms that, each for them selfs can withstand cryptanalytic attacks excellently, but when combined are garbage.

4.4.1. Socket Encryption

The socket encryption operates on the socket encryption segement (ses) of a packet given by

$$\begin{aligned} &\text{ses} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \\ &\text{ses} := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto (h_p, m, f_s) \end{aligned} \tag{61}$$

and encrypted packets are created using the following encryption functions:

$$\begin{aligned} &\text{sesEnc} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \\ &\text{sesEnc} := p \mapsto \text{enc}(\text{ses}(p), \text{seNonce}(p), \text{sockEncSK}(\text{src}(p)), \text{sockEncPK}(\text{dst}(p))) \end{aligned} \tag{62}$$

$$\begin{aligned} &\text{sesDec} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \\ &\text{sesDec} := p \mapsto \text{dec}(\text{ses}(p), \text{seNonce}(p), \text{sockEncSK}(\text{dst}(p)), \text{sockEncPK}(\text{src}(p))) \end{aligned} \tag{63}$$

$$\begin{aligned} &\text{sockEnc} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{sockEnc} := p \mapsto (\text{hdr}_6(p), \text{hdr}_n(p), \text{hdr}_{\text{s}}(p)) \mathbin{+\!\!+} \text{sesEnc}(p) \mathbin{+\!\!+} (\text{ftr}_{\text{r}}(p)) \end{aligned} \tag{64}$$

$$\begin{aligned} &\text{sockDec} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{sockDec} := p \mapsto (\text{hdr}_6(p), \text{hdr}_n(p), \text{hdr}_{\text{s}}(p)) \mathbin{+\!\!+} \text{sesDec}(p) \mathbin{+\!\!+} (\text{ftr}_{\text{r}}(p)) \end{aligned} \tag{65}$$

In the above equations a proper setup is assumed, so that the encryption/decryption algorithms can not return $\perp$. A proper implementation needs to do error/setup checking.

4.4.2. Node Encryption

Node encryption is analog to socket encryption. It just uses different keys and corresponding algorithms. It encrypts the socket segment header concealing port information to the outside world. The packets segment encrypted is called the node encryption segment nes and given by

$$\begin{aligned} &\text{nes} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_s \times \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \\ &\text{nes} := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto (h_s, h_p, m, f_s) \end{aligned} \tag{66}$$

with

$$\begin{aligned} &\text{nesEnc} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_s \times \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \\ &\text{nesEnc} := p \mapsto \text{enc}(\text{nes}(p), \text{nonce}(p), \text{nodeEncSK}(\text{srcAddr}(p)), \text{nodeEncPK}(\text{dstAddr}(p))) \end{aligned} \tag{67}$$

$$\begin{aligned} &\text{nesDec} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_s \times \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \\ &\text{nesDec} := p \mapsto \text{dec}(\text{nes}(p), \text{nonce}(p), \text{nodeEncSK}(\text{dstAddr}(p)), \text{nodeEncPK}(\text{srcAddr}(p))) \end{aligned} \tag{68}$$

$$\begin{aligned} &\text{nodeEnc} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{nodeEnc} := p \mapsto (\text{hdr}_6(p), \text{hdr}_n(p)) \; {+\!+} \text{nesEnc}(p) \; {+\!+} (\text{ftr}_{\text{r}}(p)) \end{aligned} \tag{69}$$

$$\begin{aligned} &\text{nodeDec} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{nodeDec} := p \mapsto (\text{hdr}_6(p), \text{hdr}_{\text{n}}(p)) \; {+\!+} \text{nesDec}(p) \; {+\!+} (\text{ftr}_{\text{r}}(p)) \end{aligned} \tag{70}$$

Although it would be possible, the initial source footer is not encrypted. It is initially observable in the wild as the src field of the IPv6 header and including it would therefore provide an attacking cryptanalyst with known clear-text and its position weakening the encryption.

4.5. The Datagram

The datagram is the information a packet transports from one cluster to another. To be consistent the datagram has to be equal for all nodes of sending and receiving clusters. Colloquially expressed: Datagrams are the "packets" of clusters.

Definition 4.5.1 (Datagram Constructor)**.** *A datagram is constructed as a specific form*

of a packet given by the following construction function:

$$\begin{aligned}
&\mathrm{dgrm} : \mathcal{P}_{\mathrm{psd}} \to \mathcal{P}_{\mathrm{psd}} \\
&\mathrm{dgrm} := ((v,t,f,l,n,h,s,d),h_n,h_s,h_p,m,f_s,f_i,f_r) \\
&\qquad\qquad \mapsto (\,(6,0,0, \\
&\qquad\qquad\qquad\qquad \mathit{length}(m) + N_{\mathrm{ip6}}^{\mathrm{hdr}} + N_{\mathrm{sig}}^{\mathrm{ftr}} + N_{\mathrm{i}}^{\mathrm{ftr}} + N_{\mathrm{rac}}^{\mathrm{ftr}}, \\
&\qquad\qquad\qquad\qquad \mathrm{clusterAddrOf}(s), \\
&\qquad\qquad\qquad\qquad \mathrm{clusterAddrOf}(d)\,), \\
&\qquad\qquad\qquad\qquad h_n,h_s,h_p,m,0,0)
\end{aligned} \tag{71}$$

where $N_{\mathrm{ip6}}^{\mathrm{hdr}}$ *is the IPv6 header size,* $N_{\mathrm{sig}}^{\mathrm{ftr}}$ *the signature footer size,* $N_{\mathrm{i}}^{\mathrm{ftr}}$ *the initial source footer size and* $N_{\mathrm{rac}}^{\mathrm{ftr}}$ *the rac footer size.*

Definition 4.5.2 (The set of all datagrams)**.**

$$\mathcal{D} := \mathrm{dgrm}(\mathcal{P}_{\mathrm{psd}}) \tag{72}$$

4.6. Authorized Packets

A packet is authorized if it is signed by the sending socket using the following function

$$\begin{aligned}
&\mathrm{mkSockSig} : \mathcal{P}_{\mathrm{psd}} \to \mathcal{P}_{\mathrm{psd}} \\
&\mathrm{mkSockSig} := p \mapsto (\ \mathrm{hdr}_6(p), \mathrm{hdr}_{\mathrm{n}}(p), \mathrm{hdr}_{\mathrm{s}}(p), \mathrm{hdr}_{\mathrm{p}}(p), \mathrm{payload}(p), \\
&\qquad\qquad\qquad\qquad \mathrm{mkSigFtr}(p), \mathrm{ftr}_{\mathrm{i}}(p), \mathrm{ftr}_{\mathrm{r}}(p)\ )
\end{aligned} \tag{73}$$

where

$$\begin{aligned}
&\mathrm{mkSigFtr} : \mathcal{P}_{\mathrm{psd}} \to \mathcal{F}_{\mathrm{sig}} \\
&\mathrm{mkSigFtr} := p \mapsto \mathrm{mksig}(\mathrm{sss}(\mathrm{dgrm}(p)), \mathrm{sockSigSK}(\mathrm{sgnr}(p)))
\end{aligned} \tag{74}$$

and sss is the socket signed segment given by

$$\begin{aligned}
&\mathrm{sss} : \mathcal{P}_{\mathrm{psd}} \to \mathcal{H}_6 \times \mathcal{H}_{\mathrm{n}} \times \mathcal{H}_{\mathrm{s}} \times \mathcal{H}_p \times \mathcal{P} \\
&\mathrm{sss} := (h_6,h_n,h_s,h_p,m,s,a,f_r) \mapsto (h_6,h_n,h_s,h_p,m)
\end{aligned} \tag{75}$$

4.7. Route Authenticated Packets

For the route authentication only the footer generation needs to be defined, since actual authentication is done by recomputing the footer and comparing the results with the footer given.

$$\begin{aligned}
&\mathrm{mkRacFtr} : \mathcal{P}_{\mathrm{psd}} \to \mathcal{F}_{\mathrm{rac}} \\
&\mathrm{mkRacFtr} := p \mapsto \mathrm{mkac}(\ \mathrm{ras}(\ \mathrm{dgrm}(p)), \\
&\qquad\qquad\qquad\qquad\qquad \mathrm{nonce}(p), \\
&\qquad\qquad\qquad\qquad\qquad \mathrm{routeAK}(\mathrm{nodeAK}(\mathrm{src}(p)), \mathrm{nodeAK}(\mathrm{dst}(p)))\ )\ )
\end{aligned} \tag{76}$$

where ras is the route authenticated segment given by

$$\begin{aligned} &\text{ras} : \mathcal{P}_{\text{psd}} \to \mathcal{H}_6 \times \mathcal{H}_{\text{n}} \times \mathcal{H}_{\text{s}} \times \mathcal{H}_p \times \mathcal{P} \times \mathcal{F}_{\text{sig}} \times \mathcal{F}_{\text{i}} \\ &\text{ras} := (h_6, h_n, h_s, h_p, m, f_s, f_i, f_r) \mapsto (h_6, h_n, h_s, h_p, m, f_s, f_i) \end{aligned} \tag{77}$$

4.8. The Unique Identifier

The unique identifier (uid) unambiguously identifies the datagram a packet is transporting.

Definition 4.8.1 (Set of all Unique Identifiers)**.**

$$\mathcal{U} := \text{Nat}_4 \times \text{Nat}_{20} \times \text{Nat}_{128} \times \text{Nat}_{128} \times \mathcal{H}_{\text{n}} \times \mathcal{H}_{\text{s}} \times \mathcal{H}_p \tag{78}$$

Definition 4.8.2. *The uid of a packet or datagram is given by:*

$$\begin{aligned} &\text{uid} : \mathcal{P}_{\text{psd}} \to \mathcal{U} \\ &\text{uid} := ((v, t, f, l, n, h, a_s, a_d), h_n, h_s, h_p, m, f_s, f_r) \\ &\qquad \mapsto (v, f, l, \text{clusterAddrOf}(a_s), \text{clusterAddrOf}(a_d), h_n, h_s, h_p) \end{aligned} \tag{79}$$

To access the sender and receiver adders of a datagram we define

$$\begin{aligned} &\text{sndrAddr} : \mathcal{U} \to \text{Nat}_{128} \\ &\text{sndrAddr} := (v, f, l, a_s, a_d, h_n, h_s, h_p) \mapsto a_s \end{aligned} \tag{80}$$

$$\begin{aligned} &\text{rcvrAddr} : \mathcal{U} \to \text{Nat}_{128} \\ &\text{rcvrAddr} := (v, f, l, a_s, a_d, h_n, h_s, h_p) \mapsto a_d \end{aligned} \tag{81}$$

The sender of a datagram is the set of involved sockets of the sending and receiving cluster that is

$$\begin{aligned} &\text{sndr} : \mathcal{U} \to 2^{\text{Nat}_{128} \times \text{Nat}_{64}} \\ &\text{sndr} := (v, f, l, a_s, a_d, h_n, (p_s, p_d), h_p) \mapsto \{(a, p_s) | a \in \text{clusterOf}(a_s)\} \end{aligned} \tag{82}$$

$$\begin{aligned} &\text{rcvr} : \mathcal{U} \to 2^{\text{Nat}_{128} \times \text{Nat}_{64}} \\ &\text{rcvr} := (v, f, l, a_s, a_d, h_n, (p_s, p_d), h_p) \mapsto \{(a, p_d) | a \in \text{clusterOf}(a_d)\} \end{aligned} \tag{83}$$

Sender and receiver always refer to multicast addresses of clusters, if they have assigned multi-cast addresses. It should be noted that the uid of the encrypted packet is note equal to the uid of the decrypted packet.

4.9. Initial and Final Packets

To minimize the attack surface of communicating clusters, a clusters multicast address is only used by nodes of its cluster. Packets coming from other nodes should be doped by a filter/firewall. In this regard packets are separated into two classes:

Definition 4.9.1 (Initial Packet)**.** *A packet* $p \in \mathcal{P}_{\text{psd}}$ *is a initial packet if* $\text{dstAddr}(p) \in \mathcal{N}$*, i.e. the destination is an unicast address.*

Initial packets are used for inter cluster communication.

Definition 4.9.2 (Final Packet)**.** *A packet* $p \in \mathcal{P}_{\text{psd}}$ *is a final*[16] *packet if* $\text{dstAddr}(p) \in \text{Nat}_{64} \backslash \mathcal{N}$*, i.e. the destination is a multicast address.*

Final packets are thus used fore intra cluster communication.

4.9.1. Valid Initial Packets

Having a valid signature footer does validate that a packet is sent from the signer as long as the signature algorithm or secret key is not compromised. For efficiency reasons there are some additional well formed condition added to reduce the number of valid packets as node is allowed to send. This is most simply done by defining a generator for all valid packets as reference. To do this we need the following definition:

Definition 4.9.3 (Max Valid Packet Count)**.** *Let* $u \in U$ *then the maximal valid packet count is given by*

$$N^u := \max(\#\text{sndr}(u), \#\text{rcvr}(u)) \tag{84}$$

Then the valid packet generator function is defined as

Definition 4.9.4 (Valid Initial Packet Generator)**.** *Let* $u \in U$ *then the valid initial packet generator is the function*

$$\begin{aligned} &\text{ipkt} : \mathcal{D} \times \underline{N^u} \to \mathcal{P}_{\text{psd}} \\ &\text{ipkt} := (d, k) \mapsto (\, \text{ihdr}_6^{\text{uid}(d)}(d, k), \\ &\qquad\qquad \text{hdr}_{\text{s}}(d), \\ &\qquad\qquad \text{hdr}_{\text{p}}(d), \\ &\qquad\qquad \text{payload}(d), \\ &\qquad\qquad \text{mkSigFtr}(d), \\ &\qquad\qquad \text{mkRacFtr}(\text{pktEnc}(d)\ ) \end{aligned} \tag{85}$$

[16] I could barely resit to call it terminal packet which would match aviation lingo, but that just sounds like terminal cancer.

where

$$
\begin{aligned}
&\mathrm{ihdr}_6^u : \mathcal{P}_{\mathrm{psd}} \times \underline{N^u} \to \mathcal{H}_6 \\
&\mathrm{ihdr}_6^u := (((v,t,f,l,n,h,s,d),h_n,h_s,h_p,m,f_s,f_i,f_r),k) \\
&\qquad \mapsto (\,6,0,0, \\
&\qquad\quad \mathit{length}(m) + N_{\mathrm{ip6}}^{\mathrm{hdr}} + N_{\mathrm{sig}}^{\mathrm{ftr}} + N_{\mathrm{i}}^{\mathrm{ftr}} + N_{\mathrm{rac}}^{\mathrm{ftr}}, \\
&\qquad\quad \mathrm{PSD\_PROTO\_NUM}, \\
&\qquad\quad 0, \\
&\qquad\quad \mathrm{sndr}(u)_{(k \bmod \#\mathrm{rcvr}(u))}, \\
&\qquad\quad \mathrm{rcvr}(u)_{(k \bmod \#\mathrm{sndr}(u))}\,)
\end{aligned}
\tag{86}
$$

is the initial valid header generator.

Remark. *Due to the construction of creating the rmac with encrpyted data, the actual valid initial packet data is clear text*[17]

Lemma 4.9.1 (Max Initial Packet Count)**.**

$$\forall d \in \mathcal{D} : \#\mathrm{ipkt}(d, \underline{N^u}) = N^u \tag{87}$$

Proof. Due to basic set theory is sufficient to show that

$$\mathrm{ipkt}(d,l) = \mathrm{ipkt}(d,l') \Rightarrow l = l' \ .$$

There are two cases:

1. $N^u = \#\mathrm{rcvr}(u)$: In this case the header construction yields that each packet $\mathrm{ipkt}(d,l)$ has a different $\mathrm{dst}(\mathrm{ipkt}(d,l))$ for each $l \in \underline{N^u}$, which implies the above relation.

2. $N^u = \#\mathrm{sndr}(u)$: In this case the header construction yields that each packet $\mathrm{ipkt}(d,l)$ has a different $\mathrm{src}(\mathrm{ipkt}(d,l))$ for each $l \in \underline{N^u}$, which implies the above relation.

□

Definition 4.9.5 (Set of Valid Initial Packets)**.** *The set of valid initial packets is defined by*

$$\mathcal{V} := \bigcup_{d \in \mathcal{D}} \mathrm{ipkt}(d, \underline{N^{\mathrm{uid}(d)}}) \tag{88}$$

[17] There might theoretical exceptions if something "funny" is happening with the involved crypto routines, but that will hardly happen by accident.

4.9.2. Valid Final Packets

Final valid packets are constructed form valid initial packets using the multicast packet gnerator

$$\begin{aligned} &\text{mcp} : \mathcal{P}_{\text{psd}} \to \mathcal{P}_{\text{psd}} \\ &\text{mcp} := ((v,t,f,l,n,h,a_s,a_d),h_n,h_s,h_p,m,f_s,f_i,f_r) \\ &\qquad \mapsto ((v,t,f,l,n,h,a_d,\text{clusterAddrOf}(a_d),h_n,h_s,h_p,m,f_s,f_i,f_r) \end{aligned} \tag{89}$$

Definition 4.9.6 (Set of Valid Final Packets)**.** *The set of valid final packets is defined by*

$$\mathcal{V}' := \text{mcp}(\mathcal{V}) \tag{90}$$

The above construction for neNonce yields the following, with regard to decryption, important invariant:

Corollary 4.9.1.

$$\forall p \in \mathcal{P}_{\text{psd}} : \text{dstAddr}(p) \in \mathcal{N} \Rightarrow \text{neNonce}(\text{mcp}(p)) = \text{neNonce}(p) \tag{91}$$

4.10. Acknowledgement Packets

To increase transmission reliability a sender may choose to send the same packets multiple times until either the packets deadline has elapsed or an acknowledgement in form of a packet from the destination node was received. This is considered a quasi link layer mechanism and not further discussed here. The up most bit of the IPv6 headers flow label field is used to indicate acknowledgements. The exact packet content of the acknowledgement is yet to be decided, and this section will updated when it is.

5. The PSDP Specification

The PSDP protocol facilitates the transmission of a datagram from one byzantine failure tolerant service to an other, both created by replication. It is designed to transfer the byzantine failure tolerance of the clusters to the communication between them. In temporal order the protocol has three stages the initial stage, the multicast stage and the final stage. To define the actual datagram receival sematnics and interface of the packet semantics to the interpretation theory in the appendix is required, which will be provided first.

5.1. Interfacing Interpretation Theory.

In order to apply interpretation theory it has to be decided what is the actual type of data it is to be applied on. For the formal discussion and analysis of a protocol the inputs

an outputs of the communication endpoints contain the relevant data for analysis. The input data type per socket can therefore use the following definition:

Definition 5.1.1 (The Type of Socket Inputs)**.** *Let s be a socket, then the set of all possible valid, route authenticated, decrypted, validly signed and deadline meeting inputs is given by*

$$\Omega_s^{in} := \mathcal{V} \cup \mathcal{V}' \ . \tag{92}$$

While we can reasonably assume that input elements are filtered by validity by checking the required structure of valid packets this is not so clear for the output sets. However since invalid packets get dropped eventually, its not really a restriction to model the output sets as valid input sets. Errors of invalid outputs will appear as plain loss in that model which is sufficient.

Definition 5.1.2 (Type of Node Socket Outputs)**.** *Let $s \in \mathcal{N} \times \mathrm{Nat}_{64}$ be a socket of a node*

$$\Omega_s^{\mathrm{out}} := \Omega_s^{in} \ . \tag{93}$$

in terms of discussing the output of cluster, the following definition is required:

Definition 5.1.3 (Type of cluster Socket Outputs)**.** *Let $(a, l) \in \mathcal{C} \times \mathrm{Nat}_{64}$ be a socket of a cluster*

$$\Omega_{(c,l)}^{\mathrm{out}} := \bigcup_{n \in \mathrm{clusterOf}(c)} \Omega_s^{out} \ . \tag{94}$$

This gives us the following nice corollary for the based data types of our analysis:

Corollary 5.1.1.

$$\forall (a, l) \in \mathcal{C} \times \mathrm{Nat}_{64} : \Omega_{(c,l)}^{\mathrm{out}} = \mathcal{V} \cup \mathcal{V}' \tag{95}$$

$$\forall (a, l) \in \mathcal{N} \times \mathrm{Nat}_{64} : \Omega_s^{\mathrm{in}} = \Omega_s^{out} = \mathcal{V} \cup \mathcal{V}' \tag{96}$$

In conclusion this yields a simple base type for following formal treatments:

Definition 5.1.4 (The Type of Relevant Packet Data)**.**

$$\Omega := \mathcal{V} \cup \mathcal{V}' \ . \tag{97}$$

By construction the packet interface is available on Ω and can be used to construct a sourced key value structure as required for interpretation theory:

Corollary 5.1.2 (Interpretation Theory Interface)**.** *For each cluster $c \in \mathcal{C}$ the tuple*

$$\mathcal{V}_c := \left(\Omega, \mathcal{U}, c, \mathcal{D}, \mathrm{uid}, \mathrm{dgrm}, \mathrm{sgnr}, \left\lfloor \frac{1}{3} \# c \right\rfloor + 1 \right) \tag{98}$$

is a sourced key value structure.

To elaborate on this interface the packets send and received by nodes must construct corresponding formations in the following way. If D is a set of received packets of valid the same uid, then it is physiclly avaiblie on the receving node after the deadline has passed and the formation derive the datagram, i.e. the interpretation of this packet set is simply done by

$$F := D\!\!\!/_{\mathrm{ar}_{\mathcal{V}c}}\!\!\!/_{\mathrm{sr}_{\mathcal{V}c}} \ . \tag{99}$$

More details about efficient IO handling are discussed in the later corresponding sections.

5.2. Initial Stage

The presumption of the initial stage is that each "undamaged" socket of a sending replicated program, knows the true value of some datagram d. Then, to send the datagram $d \in D$ the each socket $n \in \mathrm{sndr}(\mathrm{uid}(d))$ sends the packets:

$$\mathcal{O}_d^n := \{\, \mathrm{pktEnc}(p) \,|\, p \in \mathrm{ipkt}(d, \underline{N^u}) \wedge \mathrm{srcAddr}(p) = n \,\} \tag{100}$$

5.3. Multicast Stage

To perform the multicast stage for datagram d each socket $n' \in \mathrm{rcvr}(\mathrm{uid}(d))$ that received a packet $p \in \mathcal{P}_{\mathrm{psd}}$ checks if $p \in \mathcal{O}_d^n$ and $\mathrm{dstAddr}(p) = n'$. If all checks succeeded the node n' immediately sends $\mathrm{mcp}(p)$. It is not necessary to decrypt the packet for this stage. The set of multicast packets created this way is denoted $\mathcal{O}_d'^n$.

5.4. Final Stage

To perform the final stage to receive datagram d each node $n' \in \mathrm{rcvr}(\mathrm{uid}(d))$ that has received at least one decrypted $p' \in \mathcal{O}_d'^n$ waits until the deadline of d has passed. The time waited includes a clock synchronization error margin, that guarantees that the deadline has passed on all other receiving clusters as well. The datagram d is received by node n' if it is the interpretation of of the corresponding packet formation build using the interpretation theory interface of the sender.

5.5. Valid Setups

Let A be a set of nodes forming a cluster and $\mathrm{bft}(A)$ the number of byzantine failures the A should be able to absorb without failure then the number of nodes of a must be given by the following formula:

$$\#A = \begin{cases} 1 & \text{for } \mathrm{bft}(A) = 0 \\ 3 * 2^{\mathrm{bft}(A)-1} & \text{otherwise} \end{cases} \tag{101}$$

This implies that the failure tolerance is fixed to a third of the nodes, and nonzero tolerances are powers of two. The later may seem excessive but excludes some subtle issues, that would decrees reliability for some cluster pairs.

6. Formal Assessment

The formal security and reliability assessment laid out in the following sections is based on mathematical concepts commonly used in mathematical statistics and stochastics. The required theory is presented in a appendices on error theory and interpretation theory.

6.1. Byzantine Failure Tolerance

The above interface to interpretation theory allows the direct application of its byzantine interpretation failure tolerance theorem B.7.1 and yields a byzantine failure threshold of the protocol equal to the byzantine failure threshold of the sending cluster. However for high reliability more is required, the receiving cluster must also stay sufficiently synchronized under any circumstances, which is shown next and also the reason why the byzantine failure tolerance is set to only a third of the nodes.

6.2. Reliability

The entry point to asses reliability is also interpretation theory in particular its interpretation resilience theorem B.6.1, however its application is not direct as for the byzantine failure tolerance.

Theorem 6.2.1 (Reli)**.** *Let $u \in \mathcal{U}$ and $s = \mathrm{sndr}(u)$ be the sending node and $r = \mathrm{rcvr}(u)$ the receiving node. Further let D be the sets of received initial packets with uid u and D'_i be the final packets received by node $i \in r$ then*

$$\sum_{i \in r} \mathrm{err}^{\#}(D, D'_i) < \mathrm{bft}_{\mathcal{V}_r} \tag{102}$$

then

$$\#\{i \in \#r \,|\, \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D/_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}) = \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D'_i/_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}})\} \geqslant \left\lfloor \frac{2}{3}\#r \right\rfloor \tag{103}$$

so that $\mathrm{inter}_{\mu_{\mathcal{V}_s}}(D/_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}) = \{d\}$ *if non-empty where d is the received datagram.*

Proof. First note that $\mathrm{bft}_{\mathcal{V}_r} = \left\lfloor \frac{1}{3}\#r \right\rfloor + 1$ and let $e := \{i \in r | \mathrm{err}^{\#}(D', D_i) \neq 0\}$ then $\#e < \left\lfloor \frac{1}{3}\#r \right\rfloor + 1$. So that

$$\#(r \backslash e) = \#r - \#e > \#r - \left\lfloor \frac{1}{3}\#r \right\rfloor - 1 = \left\lfloor \frac{2}{3}\#r \right\rfloor - 1$$

an thus

$$\#(r \backslash e) > \left\lfloor \frac{2}{3}\#r \right\rfloor - 1 \Rightarrow \#(r \backslash e) \geqslant \left\lfloor \frac{2}{3}\#r \right\rfloor$$

further let it holds that

$$
\begin{aligned}
& r\backslash e \\
=&\{i \in r \,|\, \mathrm{err}^{\#}(D', D_i) = 0\} \\
=&\{i \in r \,|\, \mathrm{err}^{\#}(D', D_i) = 0 \wedge \mathrm{err}^{\mu_{\mathcal{V}_r}}(D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}, D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}) = 0\} \\
\subseteq&\{i \in r \,|\, \mathrm{err}^{\mu_{\mathcal{V}_r}}(D{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}, D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}) = 0\} \\
=&\{i \in r \,|\, \mathrm{err}^{\mu_{\mathcal{V}_r}}(D{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}, D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}) = 0 \wedge \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}) = \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}})\} \\
\subseteq&\{i \in r \,|\, \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}) = \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}})\}
\end{aligned}
$$

which finally implies

$$
\#\{i \in r \,|\, \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}}) = \mathrm{inter}_{\mu_{\mathcal{V}_s}}(D'_i{\scriptstyle /_{\mathrm{ar}_{\mathcal{V}_c}}/_{\mathrm{sr}_{\mathcal{V}_c}}})\} \geqslant \#(r\backslash e) \geqslant \left\lfloor \frac{2}{3}\#r \right\rfloor
$$

□

It should be possible to derive results with sharp bounds for the number of multicast packet losses a cluster can have without losing full synchronization. For not sufficiently corrupted initial packets, there can be further loss in the multicast stage without any affect, that is still providing full synchronization, since the threshold is only a third of the receiving clusters node count plus one. This investigation however requires to differentiate error types into corrupting and condition reducing errors and thus long and omitted in this document. The possibility to derive these sharp bounds plus some general indistinguishably considerations regarding the nodes of a cluster is also the reason for the valid setups restrictions.

6.3. DDoS Resiliency

The DDoS resiliency is not so much a direct feature of the protocol itself can be archived through the provided route authentication mechanism. The route authentication codes are constructed so that they can be checked on the encrypted packets, so that routers that have access to the required shared secret node authentication keys can filter spurious packets. In this way the number of nodes that can participate in a DDoS attack can be severely restricted making it economic feasible to provide enough computational power to make successful DDoS attacks unfeasible.

6.4. Cryptoanalytic and Quantum Computer Attack Resiliency

From the PSDP perspective these cryptanalytic and quantum attacks pose the same kind of thread. Instead of compromising a single key of an cryptographic algorithms, a single sucessful attack of that kind compromises all keys of ceratin algorithms. However the protocol does not specify the algorithms, and they can be changed if required. Further if appropriately set up the node encryption and socket encryption algorithms should be configured to belong to classes that are not susceptible to the same kind of cryptoanalytic of quantum computer attack. In that case a sucessfull attack may allow the attack to

decode at most one layer of encryption and at most get access to port information while all payload data remains encrypted.

Furthermore regarding signature algorithms, the nodes of a signing clusters should also be configured to use distinct classes of algorithms, so that in the beset case the number of successful cryptoanalytic of quantum computer attacks required to forge authenticity is equal to the senders byzantine failure tolerance[18].

In both cases a singe successful cryptoanalytic or quantum computer attack alone does not compromise an intact system and thus provides a time window to replay this algorithm while the system remain operational without any need for downtime.

In conclusion the crytoanalytic security of this protocol breaks down only in the case of a Crypto-Armageddon, where all signature algorithms[19] are revealed to be actually computational cheap to break.

6.5. Military Grade Security Capabilities and Implementations

The above assessments is build on the mathematical theory provided in the appendix, and proves that the protocol satisfies the following military application requirements:

1. Provision of minimal attack surface exposed to potentially malicious adversaries, by making cluster multicast networks cluster local.

2. Guaranteed maximal sustained damage limit proven with the above reliability theorem. Given a correct implementation a malicious adversary that is unable overcome the byzantine failure tolerance is not able to perform an action that will disable the systems ability to operate correctly.

3. Byzantine failure tolerant authentication, as proven by the byzantine interpretation failure tolerance theorem.

4. Ability to be deployed without creating any single point of failure, due to its consistent replication cluster based design.

5. Ability to be set up to be unbreakable by a single type of successful cryptanalytic or quantum computer attack.

6. Possibility to deny an adversary the use of DDoS attack by thoroughly using PSDPs route authentication mechanism.

7. Resilient to hardware loss or destruction due to consistent replication cluster based design.

8. Real-time data handling as required for several tactical data link applications

[18] The byzantine failure tolerance is the byzantine failure threshold minus one.

[19] Regarding encryption it is always possible fall back to one key pads. Thats painfully ineconomic but mathematically secure. The author is not aware of a comparable signature algorithm fall-back. Please tell him if you know one, maybe some kind of PIN-TAN equivalent - as used for online Banking?

This are only the design specifics designating this system as military grade. Polygravites implementation provides the further military grade hardening:

1. Thorough use of process and privilege separation, so that a single subsystem breach can only reveal a minimal amount of cryptographic key data.

2. Produced using a high quality test driven development process, that runs the each API test in a memory checker (valgrind), significantly reducing the probability security critical memory corruption bugs, like buffer overflows, heap overflows and double frees, etc.

3. Implemented as a portable, low latency, real-time, small memory footprint, close to the metal, C implementation.

4. Implementation priories security, reliability, and repairability over speed and still is fast.

In the following section the algorithm outlines of Polygravities implementation will be provided.

7. Low Latency, High Throughput IO-Algorithms

The PSDP protocol has been specified in the previous sections of the document. However the specification does only describe what has to be done in order to follow the protocol and not how it is done efficiently. An outline for efficient IO algorithm implementations will therefore be described in the following sections. To provide some context regarding the proposed design decisions a brief discussion of common software engineering problems associated with high throughput and low latency implementations of network protocol APIs follows:

1. The implementation is written in a slow language.

2. The implementation does not properly identify parallel subproblems.

3. The implementations parallelization carelessly uses muteness and barriers that implicitly serialize the parallel processing.

4. Archiving low latency fails due to too many dependent processing stages that need to be finished strictly in order.

5. Archiving high throughput fails due to system resource exhaustion problems. For example, in protocols such as TCP each connection of a protocol consumes at least a single kernel resource to process, exhausting the kernel resource limits fast in case of many parallel operating connections.

The proposed algorithms will, aside from the implementation language and the number of protocol stages, address of these problems. Furthermore the PSDP protocol already has the lowest logically possible number of stages, if separating the cluster local multicast network by a firewall for security reasons is a mandatory requirement. In this case there can not be less then 3 stages in such a protocol. The data has to always reach the reviving cluster, be processed by its firewall and then be exchanged over the cluster private multicast network to create a consensus[20].

7.1. Base Intervals

The basic design decision to allow parallel processing is to separate all data into batches and blocks. The non formalized notion of a batch is, that it is a collection of data having a some kind of time tags in a precisely defined time span. To make this notion precise we need to define the base interval of a cluster as a global configuration function as was done for the cluster composition and key infrastructure functions:

Definition 7.1.1 (Base Interval)**.** *The base interval of a cluster is accessed by the function*

$$\mathrm{bi} : \mathcal{C} \to \mathrm{Int}_{64} \tag{104}$$

where the codomain of bi *is the an time span in micro seconds.*

A time resolution in microseconds is sufficient since proper high availability setups should separate the nodes physically, so that a single fire, or bombing in a military context, can affect only one node of cluster at once. If at least a 300m separation is assumed for physically secure separation, then a signal moving at speed of light would already require about a micro second to travel from one node to an other[21]. Further time to process the signal data is also required. This also a practical resolution since measuring time with micro second precision as well as synchronizing clocks within that precision is not particularly difficult to implement with contemporary technology.

7.2. Batches

A batch a set of objects[22] that are processed in parallel and precisely identified through a time tag in conjunction with the base interval:

[20] One of the reasons to not add a further stage to the protocol that guarantees full synchronization in any case where the numbers of errors in the receivers multi cast network is below its byzantine failure threshold is to remain on lowest possible number of stages. The other reason is that it was estimated, that in practice preserving the low latency aspect was a more flexible design decision, since an asynchronous error correction can be added in higher protocol layers and will be required in any case for resynchronization of nodes after hardware failures.

[21] The speed of light is about $3 * 10^8 \frac{m}{s}$, so the traviling time is $\frac{3 * 10^2 m}{3 * 10^8 \frac{m}{s}} = 1 * 10^{-6} s$

[22] In the IO algorithms implementation that is usually packets and some optimally added meta data depending on the exact algorithms the batch is associated with.

Definition 7.2.1 (Batch)**.** *Let $n \in \mathbb{N}$ then $B \subset \mathcal{P}_{\text{psd}}$ is the n-th t-tag batch if*

$$\forall p \in B : (n-1)\,\text{bi}(\text{clusterOf}(n)) < t(p) \leqslant n\,\text{bi}(\text{clusterOf}(n)) \tag{105}$$

where t is a function yielding the corresponding time tag of an entry.

Since the deadlines are set by sending clusters and are by construction part of the uid, and therefore equal for all valid packets of the same datagram, deadlines allows the construction of batches of packets that are, in absence of errors equal on all receiving nodes of a cluster. However this implies a trade off that sacrifices some latency since processing of a complete batch of that kind can only start after the upper bound deadline of that batch has passed. This in case of a few input events will thus add latency in processing, however in case of massive parallel occurring input events, the ability to parallel process them will decrease average latency, so that this implies a trade-off between increasing best case latency for decreasing worst case latency, configurable by the base interval.

7.3. Blocks

Blocks are a special case of batches. They are deadline-tagged batches of datagrams that have an imposed order for logarithmic to support lookup time using the binary search [6]. In absence of sufficient errors blocks are, like the contained datagrams, equal on all nodes of a cluster and can be combined into structures like blockchains and/or prefix trees to add additional features like modification resilience and constant element lookup times[23]. The Eidetic Datagram Protocol described in an other document provides exactly these features for the PSDP data input of distributed applications.

7.4. Deadline Queues

Deadline queues are the basic concurrent real-time data structure that is used to implement the IO algorithms described below. It consists of two processes, one for queuing new packets and one for removing batches of deadlines that have passed. In more detail, the queue is given by the following data:

1. A buffer of deadline-tagged batches.
2. A reference into the buffer pointing to the oldest batch.

Furthermore the deadline queue a provides the following API:

1. Removing the oldest batch, after waiting until its deadline plus some clock synchronization error margin has passed.

[23] A prefix tree using the datagram deadline will have finite maximal depth and thus provide constant loookup time for batches. The base interval in conjunction with maximal input bandwidth of the hardware yields a fixed upper limit for possible input batch sizes. Together this yields a strict constant upper bound for lookup complexity, resulting in overall constant lookup times.

2. Adding an entry to a batch or drop it if the deadline plus some clock synchronization error margin has passed. This implicitly creates new batches as required.

Actual implementation need to handle a lot more details, like for example, handling makeups after host has gone to sleep and general supporting for waiting/sleeping for power need reduction. The above is just the core of what functionality a deadline queue must provide in oder to implement the IO algorithms outlined next. However before these are discussed the limitations induced by this approach should be discussed:

7.5. Limitations

While the above deadline queues yield a simple[24] way to create batches of packets whose packets can be processed in parallel, it also induces the following practical limitations

1. Potentially unlimited memory requirement, if deadlines may ba arbitrarily far in the future.

2. Potentially unlimited cpu time requirement, if deadlines may be arbitrarily far in the future.

3. Potentially unstable and chaotic system behavior as often observed in dynamic systems with memory/backlogs.

The reason for for all of these limitations is simple: The farer away in the future a deadline is, he more time there is to put packets into the same batch of a deadline queue. When the deadline of such a batch passes, the system will try to process all its packets at once in parallel, possibly exhausting all system resources. Allowing deadlines arbitrarily far away in the future therefore allows a malicious adversary to perform a simple DoS attack[25]. This is of course not tolerable for an actual implementation but can be mitigated by introducing following constraint:

- There must be an earliest possible acceptance time for packets given by the deadline minus a receiver specific timeout[26]. Packets received earlier must be dropped to mitigate all of the above problems.

In this way the number of batches in the queue, as well as the number of packets in one batch has a well defined upper limit given by the hardwares input bandwidth. In practice this implies that the low latency property of the following algorithms is really feature like but a restriction on the allowed deadlines, configurations and usable hardware.

[24] That is simple aside from the involved concurrency handling, which never should be assumed to be simple.

[25] The the authentication and authorization features will clearly identify where such an attack came from. However peer nodes may have been compromised, so that assuming that no identifiable pair would be stupid enough to start a DoS attack is a fallacy.

[26] Thats more a "timein" in this case.

7.6. Computing Interpretation Theory Formations for Packets

Given the above deadline queues an experienced software engineer should already have a hunch on how to implement the IO-Algorithms. However it has not jet been discussed how to efficiently build the interpretation theory formations of packtes needed by the final protocol stage to determine the received datagrams. In this section, it will be shown that building these formations this can be done in quasi linear runtime complexity. With regard to this first note the following:

Corollary 7.6.1. *If there is a packet of a specific uid in a batch of a deadline queue, then, once the batches deadline has passed, it contains all of the valid non-timed out packets of the same uid and thus provides the required base data to build an interpretation theory formation of packets for that uid.*

If D is the data of such a batch with packets from cluster c then $D/_{\mathrm{ar}_{\mathcal{V}_c}/\mathrm{sr}_{\mathcal{V}_c}/\mathrm{sk}_{\mathcal{V}_c}}$ is the set of all formations to investigate for potential interpretations as datagrams, derived from D.

That, at a first glance, may not look not very promising with respect to runtime complexity, since it requires the creation of a representation of nested equivalence classes. However creating this representation is computationally cheap as the following analysis reveals:

1. For the key, in this case the uid, of a source key value structure it holds that
   a) All formations have the same key.
   b) All significations have the same key.
   c) All Associations have the the same key.
2. For all significations of a formation it holds that
   a) They have the same key.
   b) They have the same values.

which is the structure of blocks occurring for arrays sorted by an compatible order on significations. To be precise, the following order will allow to construct formations of an array of significations by simply applying a sorting algorithm:

Definition 7.6.1 (The Signifying Order)**.** *Let $\mathcal{V}_c$ be the sourced key value structure of packets from cluster c that its signifying order is by definition*

$$p \preceq q :\Leftrightarrow \quad (\mathrm{key}_{\mathcal{V}_c}(p) \leqslant \mathrm{key}_{\mathcal{V}_c}(q)) \vee (\mathrm{key}_{\mathcal{V}_c}(p) = \mathrm{key}_{\mathcal{V}_c}(q) \wedge \mathrm{val}_{\mathcal{V}_c}(p) \leqslant \mathrm{val}_{\mathcal{V}_c}(q)) \quad (106)$$

where the keys the values are interpreted as natural numbers to give them an order.

To apply this order, associations can be created by soring packets by association reference and since associations with more then one packet have significance 0 they can be filtered out and the above ordering can then be directly applied to an array packets. This array sorted by an order that entails the signifying order will therefore yield formations

as subarrays of the same key, where the significations of these formations are subarrays of the formation subarrays. Due to the fact that are many sorting algorithms with worst case quasi linear complexity, that is $\#D * \log(\#D)$ [6] and the fact that filtering[27] has linear complexity, creating formations has quasi linear complexity. Also there is radix sort with linear complexity that might be adaptable, to gain strict linear complexity. In conclusion this indicates that the computationally expensive part of the IO algorithms are the cryptographic routines and finally it should be noted that:

Corollary 7.6.2. *The signifying order is the same for all clusters.*

[27] That is not meat to imply that you can not implement filtering with worse complexity. Just do it properly.

7.7. An Output Algorithm

First collect datagrams to be send at once into a batch $\mathcal{O}_0$ and perform the following steps:

1. Sort the datagrams by sender to access persistent key data once per sender. Sign each packet in $\mathcal{O}_0$ in parallel and put the results into a batch $\mathcal{O}_1$.

2. Sort the datagrams in O_1 by source and destination socket[28] to access required persistent key data for socket encryption once per source destination socket pair. Encrypt the socket encryption segment of each entry in $\mathcal{O}_1$ in parallel and put the results into a batch $\mathcal{O}_2$.

3. Sort each entry in $\mathcal{O}_2$ by source and destination addresses, to access required persistent key data once per source destination node pair. Encrypt the node encryption segment of each entry in $\mathcal{O}_2$ in parallel and store the results in $\mathcal{O}_4$.

4. Create the valid packets form $\mathcal{O}_4$ using the ipkt function in parallel and put them into a batch $\mathcal{O}_5$.

5. Sort each entry in $\mathcal{O}_5$ by source and destination addresses, to access the persistent node authentications key data once per source destination pair. Compute the route authentication codes of each packet in $\mathcal{O}_5$ and append them in parallel. Store the result in $\mathcal{O}_6$.

6. Send all packets in $\mathcal{O}_6$ though the corresponding network interfaces, then put it into a deadline queue for to resend them until acknowledgement packets have been received or the deadlines timed out.

To implement parallelism of the above algorithm all that is needed is a simple map-reduce framework, so that using massive parallel co-processors, like Nvidias Tesla Series, or Intels phi Series does not pose a principal engineering challenge - aside from implementing the required cryptographic algorithms secure and sound on these architectures. Furthermore, a proper implementation will do privilege separation for each of the stages so that at most cryptographic keys of a single stage can be compromised by a single security breach and a proper implementation will only require a fixed amount of system resources like file descriptors for each batch. Since the number of batches has an upper bound due to the low latency requirement, a sufficiently dimension system with properly designed replicated applications will thus never run out of system resources. independent of the number of parallel handled persistent socket datagram connections. The same holds for the Input algorithm that will be discussed next.

[28]Note that all nodes of a cluster are required to used the same encryptions keys, as discussed above.

7.8. An Input Algorithm

The input algorithm is centered around a deadline queue and thus has two main processes, one filling the queue, called the collector process, the other emptying the queue and providing the designated replicated applications with their decrypted and authenticated datagrams, called the dispatcher process.

7.8.1. The Collector Process:

The collector process begins with receiving data form the interfaces of a host and puts them into receivable time tagged batches of packets. Note that the collector process includes routing logic, so that the kernel may not, depending on the exact configurations and setups, simply route PSDP packets. For each such batch I_0 received the following processing stages are performed:

1. Wait until the latest possible receival time of $\mathcal{I}_0$ plus a clock error safety margin has passed, so that no new packets can be added.

2. Sort $\mathcal{I}_0$ by source and destination addresses to access persistent authentication key data once per source destination pair. Check the route authentication codes for each packet in $\mathcal{I}_0$ in parallel, drop all non authenticatable packets and put the results into a batch $\mathcal{I}_1$. If no key information for the route is available its up to the system administrator to decide weather packets with non-local destinations ought be dropped or not.

3. Filter $\mathcal{I}_1$ for packets with local destination and these put these into $\mathcal{I}_2$, send the non-local not-dropped packets back to the host interfaces for routing them forward.

4. Create a batch of acknowledgement packets for $\mathcal{I}_2$ in parallel and send it out the interfaces.

5. Filter $\mathcal{I}_2$ for final packets and put them into a batch $\mathcal{I}_3$ in parallel. Create a batch of final packets from the initial packets $\mathcal{I}_2 \backslash \mathcal{I}_3$ using the mcp function in parallel and send them out over the local clusters dedicated multicast network using an deadline queue to repeat sending each final packet until either its deadline expired or an acknowledgement packet was received.

6. Sort I_3 by source and destination addresses to access persistent node encryption key data once per source destination pair. Decrypt all packets in I_3 in parallel and put the results into a batch $\mathcal{I}_4$,

7. Sort I_4 by source and destination socket to access persistent socket encryption key data once per source destination pair. Decrypt all packets in I_4 in parallel and put the results into a batch $\mathcal{I}_5$,

8. Sort I_5 by source, to access persistent signature key data once per source. Verify all packets signatures in $\mathcal{I}_5$ in parallel dropping all packets whose sources can not be validated and put the results into a batch I_6.

9. Investigate the deadlines of each packet in I_6 and put them into the respective batches of a input deadline queue if the deadline of the packet plus an clock error margin has not passed. Note that the deadline of a packet may be much later then the receival time tags of the batches processed in the stages before.

In this case the stages are also properly implemented using privilege separation, as for the output algorithm.

7.8.2. The Dispatcher Process

The fundamental purpose of the dispatcher process is to apply interpretation theory to derive the received datagrams and forward these in batches to their corresponding destination sockets. In contrast to the collector process the dispatcher process operates on deadline tagged batches, so that its processing of a received packet might start a significant time after it was processed by the collector process. The dispatcher process has the following stages:

1. Wait until the input deadline queues oldest batch deadline plus a clock error safety margin has passed remove it and put it into a batch $\mathcal{D}_0$.

2. Sort $\mathcal{D}_0$ by association relation, and put the results into $\mathcal{D}_1$.

3. Filter $\mathcal{D}_1$ for relevant associations and save the results in $\mathcal{D}_2$. Relevant associations have one packet per association, so that $\mathcal{D}_2$ is a simple array of packets, each representing a relevant association

4. Sort the $\mathcal{D}_3$ using a block order for fast packet lookup that entails an ordering by the signifying order, and put the results in D_4.

5. Find the corresponding block boundaries of the signifying oder and create a representation of the formations in $\mathcal{D}_4$ and put them into $\mathcal{F}_1$.

6. Evaluate the significance measure of each signification in each formation $F \in \mathcal{F}_1$ by simply counting packets in them. Put the result in a formation representation $\mathcal{F}_2$

7. Compute the cleared formation of each formation $F \in \mathcal{F}_2$ by filtering them using the computed significance levels and put the result to a cleared formations representation $\mathcal{F}_3$.

8. Drop all cleared formations in $\mathcal{F}_3$ that contain more then one signification and thus have no interpretation.

9. Calculate the interpretations, that is received datagrams, in $\mathcal{F}_3$, put into a batch. Put all signature validation data that is hashes and signatures into a second batch path pass both batches to the destination socket.

All of the above filtering can be done in quasi linear time, an in part parallel. Furthermore the separate block containing validation data can be used to create hashes for linking the dispatchers output blocks into cluster local blochchains. For this purpose Polygravity has developed a higher level protocol called the Eidetic Datagram Protocol that will create interlocked cluster local blockchains making it defacto impossible for clusters to lie about their history of received packets without being detected - even after signature keys may have been compromised.

7.9. Scalability Analysis

In this section we discuss some scalability results with respect the above algorithms. The protocol itself as a protocol no practical scalability restrictions since a IPv6 address space combined with 64-Bit port numbers is sufficient to never run into practical restrictions if at least a minimum of care is taken with regard to resource management.

The actual scalability discussion thus must focus on the relation between the number of nodes a cluster consists of and its bandwidth, latency and resource requirements There some mayor configurable trade-offs with respect to that:

1. Parallel throughput versus low latency
2. High byzantine failure tolerance versus low bandwidth footprint.
3. Cryptographic security versus low latency.
4. Later deadlines versus resource requirements.

In the following this will be discussed in more detail. In principal the protocol is usable by almost any kind of hardware, but not every kind of hardware is able to support the same levels of security, throughput and latency. When dimensioning hardware for an application the following trade-offs must therefore be carefully considered.

7.9.1. Throughput vs. Latency

The above IO-Algorithm design already priories throughput over latency by using base internals to group packets into parallel processed batches. In principle these intervals can be set arbitrarily short. However the use of several stages in the algorithms implies that this will serialize the processing to a high degree, so that in situations that have a lot of parallel events such a minimal latency configuration inhibits the ability use available parallel processing power to full extend and therefore increases average latency.

7.9.2. Byzantine Failure Tolerance vs. Bandwidth

The most obvious trade-off is probably the one between the required byzantine failure tolerance and available/affordable bandwidth. For this it must be considered that an n-Node cluster should have n physically different outgoing network connections that route to destinations over different hardware. While any byzantine failure tolerance threshold

can be reached in principle, implementing it will become very expensive fast. In that regard clusters 1,3,6 and 12 nodes will primarily be deployed. Larger numbers of nodes may be used in some situations but will be very rare.

To see the reasons for this, a formula for the bandwidth requirement of the cluster local multicast network is given next. Let n be the number of Nodes of a cluster and Δd_{uc} its nodes main network interface maximal bandwidth, when when these interface is under full load from receiving initial packets on full load, the cluster local multicast network bandwidth required for congestion free multicast handling Δd_{mc} must satisfy

$$\Delta d_{\mathrm{mc}} \geqslant n * \Delta d_{\mathrm{uc}} \ . \tag{107}$$

Furthermore to get proper hardware failure tolerance the multicast network should also be build using n different physical networks.This can be done by giving each node its dedicated multicast network to sends its final packets on. In that case the number of network of connectors N_{nc} and directly associated hardware requirements scales with

$$N_{\mathrm{nc}} = n^2 \ . \tag{108}$$

As one might have expected, some nonlinear scaling factor in this system. However using IPv6 multicast technology removes this factor from the multicast network bandwidth requirement and shifts it to hardware components that, depending on the actual technology used can be cheaply mass produced[29]. Higher multicast network bandwidth requires more processing power as well. However on most hardware processing power is in relation so oversized that this may not by that big of an factor in many cases. If it is a factor, the number of cores per node of a cluster should be equal to the number of nodes of that cluster, as rule of thump.

7.9.3. Cryptographic Security vs Latency

The PSDP protocol allows complete configurable control over the cryptographic algorithms used for encryption, signing and route authentication. It is therefore possible to reduce latency by deploying computationally cheaper algorithms, that usually are less secure, however[30].

7.9.4. Later Deadlines vs. Resource Requirements

By later deadlines it is meant that the time window in which packets will be received and is configured to be larger. If the system as sufficient resources to manage the required deadline queue sizes this is no problem. However great care should be taken when estimating this. Bigger packet acceptance time windows imply bigger maximal sizes of

[29] The basic network technology should be fast, simple and stupid technology that can be formally verified. A network hardware monoculture will break byzantine failure tolerance in case of security critical bugs.

[30] There algorithms that just are better in that respect, but to avoid mono-cultures and reach high levels of byzantine failure tolerance regarding analytic and quantum computer attacks lower tier algorithm classes may need to be deployed too.

datagram batches, that may be dispatched at once to an application, an open attack surface or intentional or accidental DoS attacks. Mind your system configurations and dimensioning carefully otherwise doom will follow.

7.10. Early Benchmark Results

Polygravities implementation of this protocol aims at providing a flexible maximum security implementation usable on low processing power hardware as good as possible. In fact, the first product line will be an automatic transaction settling system for the Internet of things (IoT). This application case is characterized by a massive numbers of cheap and computational weak Internet of Things clients, communicating with high performance accounting services, responsible for handling the potentially massive amounts fo parallel occurring transactions triggered by the cheap IoT client devices. To test the ability of implementation to satisfy the corresponding performance requirements an benchmark was performed on an old iMac with an intel i7 870 processor[31] that has 4 cores. For this benchmark the following cryptographic setup was used:

Route Authentication Algorithm A Poly-1305 XSalsa20 combination.

Node Encryption Key exchange with X25519 and encryption with XSalsa20.

Socket Encryption Key exchange with X25519 and encryption with XSalsa20 .

Socket Authorization Ed25519 digital signatures on 512bit BLAKE2b hashes.

The above selection of algorithm is a place holder selection to simulate computational load for fast very high security algorithm selections. The implementations where provided by libsodium which is based on work of Daniel J. Bernstein et al, who, to the knowledge the author, has an excellent track record in providing secure and solid cryptographic systems. With this setup we reached a throughput of about 12000 datagrams, that is 3000 datagrams per 7-jear old core. Further benchmark details can be seen in the fig. 2 and fig. 3 . It should be noted, that Polygravities PSDP stack implementation priorities security over speed and uses full privilege separation. Each processing stage is started as an separate system process with access rights to only the required cryptographic key information and drops drops these rights as soon as the keys are read, before processing actual input. In that regard this benchmark is very conservative and has ample space for further improvements but it demonstrates the stated quasi linear complexity of the IO-Algorithms clearly.

8. Conclusion and Outlook

In the above sections the PSDP datagram protocol has been formally specified and its security and reliability have been formally proven. Fellow researchers should have noted the slight sales pitch of declaring restricting properties with feature attributes that

[31] This processor was released in 2009!

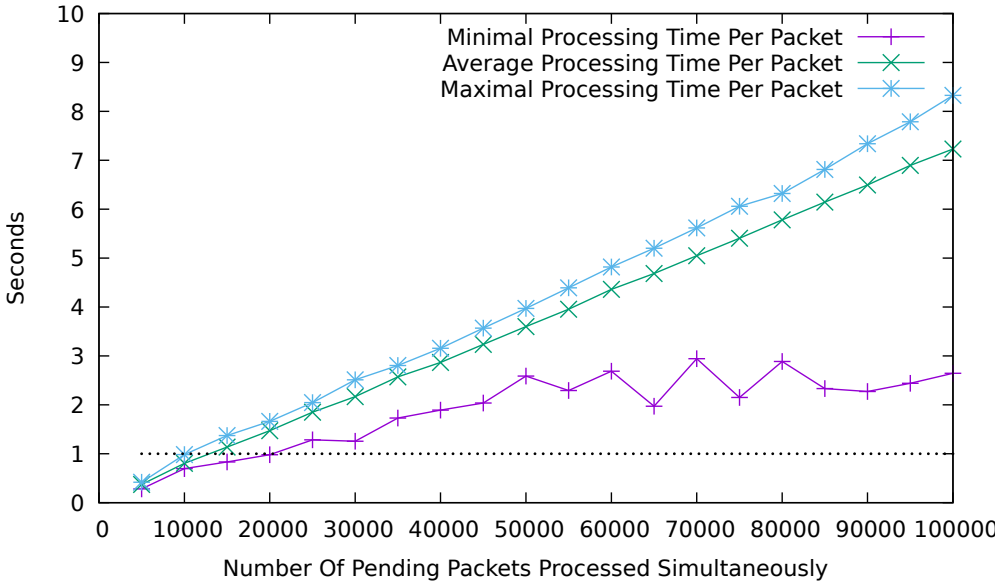

Figure 2: Processing time for packets pending in a single batch.

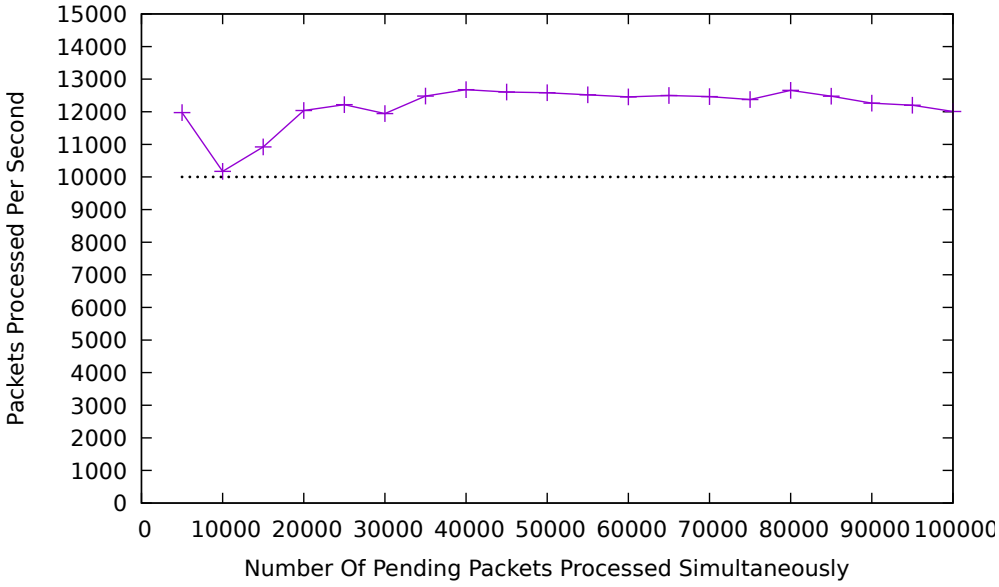

Figure 3: Number of processed packets in a single batch in relation to total batch size.

sound like features, most notable real time and low latency support. Also its all true. Polygravity, as a private research based enterprise needs to signal to its investors[32] that we are very well able to sell our products as well as doing the required down to earth research and development. In that regard a compromise has to be found between marking pitch and totally detached objective analysis and description. In that regard the later had a bit too much wight in the previous sections, so that the following contents show already started, planed or possible product developments based on the Polygravities PSDP implementation

8.1. Civil Applications

The motivating primary objective for the design of this protocol has been to develop a backbone for a real time transaction settling system and depending applications. However it has much more application scope that will be discussed thereafter.

8.1.1. Automated Transaction Processing

For automated transaction processing solutions it is necessary to provide the highest levels of security and reliability possible under given application constraints. The PSDP protocol provides all the security needs and flexibility regarding authentication, authorization and encryption requirements for that application domain. Furthermore, the described IO algorithms make it simple to build interlocked cluster local blockchains, as done by Polygravities Eidetic Datagram Protocol, that practically guarantee the detection of tempering with stored transactions, while leaving the actual transaction data local on a cluster. Furthermore having the data local, allows to delete old blocks, and to satisfy regulations that limit time transaction data may be saved. In this regard Polygravitys PSDP Protocol innovation and implementation solves the most common and pressing practical problems that contemporary blockchain technologies have. To that end the solutions are described in more details in the documentation and specification of the Eidetic Datagram Protocol as well as the Eidetic Accounting protocol.

8.1.2. Digital Issuing of Fiat Money

The demand analysis of Polygravity found, that the current hype around crypto currencies, is not so much a demand on currencies them selfs, but for some key features that they aim to provide. In that regard Polygravity estimated that implementing the logic required to issue common types of Fiat money is a sufficient market positioning[33]. The fiat moneys underlaying fundamental bookkeeping system, aka double entry bookkeeping is around for hundreds of years, and thus is proven to be very robust and scalable.

[32] If you are an investor and for whatever reason , do not like what we are doing, but plan to invest to get leverage to sabotage us, read the appendices and the military applications following carefully between the lines. Then, in reference to well known Dirty Harry line, ask yourself: Do I feel lucky?

[33] Following mathematical best practices, this is not an understatement. Understatement and humor in general are absolutely out of place in mathematical treatments as thoroughly discussed in the Article on Humor in the mathematics lexicon [7].

Doing Bookkeeping by hand is just incomparably more error prone then digital bookkeeping. Of course these old systems are not unconditionally stable and there are plenty of historical examples for failures. Naive digitalization would provide automated attacks and frauds and thus amplify the old systems problems. However, instead of issuing new types of Money, Polygravities research found that the available blockchain technology components can be used to mitigate the problems of these tried and tested systems to issue money and consequently developed an transaction system, that is able to satisfy central bank security and reliability requirements. This is also the explanation for the military grade security design and hardening of our system, and the primary reason for the fully mathematical protocol specification and assessment above. In war the currency system is, and has historically been, on of the main targets to weaken the enemy state, or more precisely weaken its ability to keep its war-machine financed. For a central back infrastructure it must therefore be assumed that it will be attacked with the full arsenal of military information warfare technologies. This thread level analysis has been considered and included in the PSDP protocols base design decisions and Polytravities implementation. However the automated transaction system alone is necessary but not sufficient to provide a completely digital fiat money issuing and managing system and Polygravity is looking for Partners to develop and remaining required system components.

8.1.3. General Automatic Market Implementations

Using automated, low latency transaction settling to implementing markets, is itself not a new Idea. But as to the knowledge of the author Polygravities system is the first protocol based inherently decentralized approach to this problem.

8.1.4. Network Stabilizing Renewable Energy Markets

The contemporary massive deployment of renewable energy sources poses great technical challenges in maintaining network stability, due to the fact that power gain and loss becomes irregular in time an place. To stabilize the power network it is thus necessary to drain or add power at specific power network nodes at specific times. For this control problem reaction times in the milisecond resolution are required. A sufficiently advanced implementation of Polygravities automated transaction settling protocol build on top of the PSDP protocol will be able to provide transactions settling speeds fast enough to design local electric power markets that stabilize the grid using market prizing. This will further give investors good estimates on what kind of storage power plant technology investment is profitable at what places. To provide stable power networks be such means still needs a proper regulatory legal framework, however it will be a huge step forward, in comparison to contemporary rather clumsy, but successful, regulatory frameworks, like the German "Erneuerbare-Energien-Gesetzt". Furthermore due to the security implications of insecure power network control, these real-time markets need very high levels of security that can be provided by Polygravities solutions.

8.1.5. Confidential Voice and Video Peer to Peer Communications

This is an initially non intended but in fact obvious application case of the protocol. Only the data encoding needs to be specified and the corresponding frontend applications be written.

8.1.6. Confidential Voice and Video Chatrooms

This application in combination with the above payment system would be very interesting for investors from the adult entertainment industry, providing easy virtual strip clubs with a fair trade-off between legal operation and payment as well as user and provider confidentiality.

8.2. Military Applications

For military applications some additional features, that in a civil context are uneconomic may be mandatory. The most noteworthy seem to be the following:

1. Two stage node encryption systems consisting of two staged encryption algorithms of different kind to provide resilience against analytic and quantum computer attacks on the node encryption level and even keep port information secret in case of a single successful attack.

2. White Noise Stenography, that is permanently sending random data between nodes, even if no actual messages are send. Proper use of random nonces will make all transmitted data aside from the necessary IPv6 header data like white noise. Allays sending data will make it impossible to correlate packet traffic data with actual communication events.

3. Plausible Deniability Based Stenography: Hide the actual data stream in an other data stream that has a plausible non related and unsupicious function.

8.2.1. Secure Authenticated, Encrypted and Deniable Communications

This application is the civil communications solutions with added stenography.

In case of white noise stenography in potentially provides the full hardware bandwidth and is thus usable to connect military equipment like air surveillance systems at different geo-locations, making testing the response time of these systems by accidental violating some nations air space inconclusive. Furthermore, such a technology provides capabilities to perform a deceptions regarding the systems detection capabilities and reaction times. Last the white noise signal gives a simple heartbeat signal proving that at least something is still working. The heartbeat functionality can of course be extended to provide more status details.

Plausible deniability based stenography severely limits the usable bandwidth. Most the bandwidth will be used for the cover data. However it might still be an invaluable tool for covert operators, that like so may people have the habit of sharing pictures over

Internet or just likes to hang around in the virtual strip rooms provided by the civil application above. Also this security level for communications might be of interest in some civil applications, presumably big companies, as well.

8.2.2. Command and Control Systems

The implementation of command and control systems using Polygravities base technologies makes providing the following mandatory features easy:

1. Usability of the most secure encryption methods known.
2. Usability of the most secure digital signature algorithms known.
3. Usability of the most suitable known authentication codes to mitigate against DDoS attacks.
4. Byzantine Failure Tolerance, so that single more simultaneous security breaches do not require the system taken off operation and repaired and mitigated during operation.
5. DDoS resiliency.
6. Immutability of stored information, using the Eidetic Datagram Protocol as base.
7. Complete support of data compartmentalization, also using the Eidetic datagram protocol as base. The persistent socket datagram protocol provides interlocked blockchains, thus practical immutability of data. However a capture of a system by an adversary will just reveal data stored on the captured cluster in addition to some cryptographic meta data. The meta data does not allow to deduce further information, however it may be used to verify other data provided the adversary gathered it by other means. It is not worthless, but with proper deployment information compartmentalization is guaranteed to remains intact.
8. Complete support of deleting old data. The Eidetic Datagram Protocol makes data unmodifiable not undeleteable.
9. Deniable communications as discussed above.
10. Heartbeat and automatic status reports as mentioned above.

It should however be noted, that a sound security systems design must not only take care of the communications protocol and data storage problems, as Polygravities technology solves, but of every system component from low level hardware, over the user interfaces, up to the overall organizational structure.

8.2.3. Tactical Data Links

Tactical data links are a huge class of military grade subsystems and depending on the actual purpose of the link the following features are easily provide by Polygravities technology:

1. Real-time support. In principle even hard real-time support[34]
2. Support for cheap hardware due to our initial commitment to provide automated transaction settling services for the emerging IoT markets. For example, this allows to deploy a bunch of "dirt" cheap drones as relay clusters for tactical data link traffic.

Tactical data links can of course be seen as a part of an encompassing command and control systems so that the points mentioned there also apply.

[34] Providing hard real-time, as in mathematically proven reaction time guarantees, is not a priority of Ploygravities current R&D operations, since it requires hard real-time guarantees of operating systmems as base. These guarantees are not provided by common standard operating systems. However there is no fundamental limit that prevents satisfying hard real-time requirements with the above protocol and algorithms.

A. Error Theory

In this section a little theory for discussing practically any kind of is-ought mismatch errors .

A.1. Results

To give formalism its due diligence we first need the following seed meta definition:

Meta Definition A.1.1. *Set Of Possible Results*[35]

$$\Omega := \{r|\ r \textit{ is possible in investiation context } \} \tag{109}$$

Possible results could be a lot of things, like tangible products from a production line, a set of requirements for a product development, or as used above, packets in input or output sets of a socket.

A.2. Outcomes

Multiple results yield an outcome, that is modeled the same way as events statistic and stochastic modeling, since events have the same mathematical structure as outcomes:

Definition A.2.1 (The Set of Possible Outcomes)**.** *A set of outcomes $\mathfrak{O}$ for the possible results set Ω is a σ-algebra over Ω.*

Remark. *The appropriate construction of σ-Algebras is analog to the ones used for stochastic and statistic modeling and not further discussed here. For finite countable sets, like for the protocol investigation in the σ-Algebra can simply be given by the power set, i.e.:*

$$\mathfrak{O} = 2^{\mathfrak{R}} \tag{110}$$

which is also called the discrete σ-Algebra.

Remark (In General Outcomes are not Events)**.** *An event in statistic and stochastic modeling corresponds to alternatively occurring observations of a single random process, while an outcome is a set of simultaneously occurring observations of a set of not necessarily distinct processes. If an event corresponds to simultaneous execution of the same random process, like simultaneously throwing multiple dice of the same type, then it is an outcome.*

A.2.1. Targets

The concept of error is only meaning full relative to some target. While the target is often as deceptively simple as "the truth", it more generally can be defined as collection of prohibited and mandatory results.

[35] This could be called observation space , make the use of "Omega" as symbol natural.

Definition A.2.2 (Target)**.** *Let $\mathfrak{O}$ be a σ-Algebra, then a tuple $T = (P, M) \in \mathfrak{O}^2$ is a target if*

$$P \cap M = \varnothing \tag{111}$$

where $P = T_0$ is the set of prohibited and $M = T_1$ is the set of mandatory results results.

A.2.2. Deficient Outcome

Definition A.2.3 (Set of Deficient Outcomes)**.** *Let $T \in \mathfrak{O}^2$ be a target, then the set of deficient outcomes is*

$$\mathfrak{D}(T) := \{O \in \mathfrak{O} | O \cap T_0 \neq \varnothing\} \tag{112}$$

A.2.3. Sufficient Outcome

Definition A.2.4 (Set of Sufficient Outcomes)**.** *Let $T \in \mathfrak{O}^2$ be a target, then the set of sufficient outcomes is*

$$\mathfrak{S}(T) := \{O \in \mathfrak{O} | O \notin \mathfrak{D}(T) \wedge T_1 \subseteq O\} \tag{113}$$

A.2.4. Outcome Measures

Relevant outcomes are defined by constructing a relevancy measure.

Definition A.2.5 (Outcome Measure)**.** *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space an $T \in \mathfrak{O}^2$ be a target, then μ is an outcome measure of T if*

$$\forall O \in \mathfrak{D}(T) : \mu(O) > 0 \tag{114}$$

$$\forall O \in \mathfrak{S}(T) : \mu(O) > 0 \tag{115}$$

Definition A.2.6 (Relevant Outcomes)**.** *An outcome $O \in \mathfrak{O}$ is relevant with respect to an outcome measure μ if $\mu(O) > 0$.*

Definition A.2.7 (Irrelevant Outcomes)**.** *An outcome $O \in \mathfrak{O}$ is irrelevant with respect to an outcome measure μ if $\mu(O) = 0$.*

A.2.5. The Principal Counting Outcome Measure

If the set of outcomes is countable, as in the above protocol analysis, the following is measure not the only measure usable or used but the principal one, as will be explained by its semantic analysis later:

Definition A.2.8 (Principal Counting Outcome Measure)**.** *Let $T \in \mathfrak{O}^2$ be a target, then the principal Counting Outcome measure is defined as*

$$\mu_T^{\#} := O \mapsto \#(O \cap (T_0 \cup T_1)) \tag{116}$$

Lemma A.2.1. *The principal countable output measure is indeed a measure.*

Proof. Non-negativity and having a null empty set are directly inherited from the counting measure. The σ-additivity follows as follows: Let $\{O_i\}_{i\in\mathbb{N}}]$ be a family of pairwise disjunct outcomes, then

$$\begin{aligned}\mu_T^{\#}\left(\bigcup_{i\in\mathbb{N}} O_i\right) &= \#\left(\left(\bigcup_{i\in\mathbb{N}} O_i\right)\cap(T_0\cup T_1)\right)\\ &= \#\left(\bigcup_{i\in\mathbb{N}}(\mathcal{O}_i\cap(T_0\cup T_1))\right)\\ &= \sum_{i\in\mathbb{N}}\#\left(O_i\cap(T_0\cup T_1)\right)\\ &= \sum_{i\in\mathbb{N}}\mu_T^{\#}\left(O_i\right)\end{aligned}$$

and $\mu_T^{\#}$ is thus sigma additive too and therefore a measure. □

Corollary A.2.1. *For the principal countable output measure all non mandatory and non prohibited results are irrelevant, since*

$$\mu_T^{\#}((T_0\cup T_1)^{\complement}) = 0\ . \tag{117}$$

A.3. Error Measurement

In the above section results and outcomes where categorized with respect to objectives given by the context the error theory is applied in, however the actual error is still not defined. To do this we start with defining additional important distinctions:

A.3.1. The Loss Definition

Definition A.3.1 (Loss)**.** *Given an target $T\in\mathfrak{O}^2$ and an outcome $O\in\mathfrak{O}$ the loss in O relative to target T is defined as*

$$\mathrm{loss}_T(O) := T_1\backslash O \tag{118}$$

A.3.2. The Excess Definition

Definition A.3.2 (Excess)**.** *Given an target $T\in\mathfrak{O}^2$, and outcome $O\in\mathfrak{O}$ the excess in O relative to target T is defined as*

$$\mathrm{excess}_T(O) := O\backslash T_1 \tag{119}$$

Corollary A.3.1. *Not all excess is necessarily prohibited.*

A.3.3. The Error Definition

With the above distinctions the error can naturally be defined as the sum of the measured lost and the measured excess:

Definition A.3.3 (Error Relative To Target)**.** *Let* $(\Omega, \mathfrak{O}, \mu)$ *be a measure space and* $T \in \mathfrak{O}^2$ *be a target then*

$$\mathrm{err}_T^\mu(O) := \mu(\mathrm{loss}_T(O)) + \mu(\mathrm{excess}_T(O)) \ . \tag{120}$$

However the error can be defined on any kind of measure space:

Definition A.3.4 (Error)**.** *Let* $(\Omega, \mathfrak{O}, \mu)$ *be a measure space then*

$$\mathrm{err}^\mu(M, O) := \mu(O \backslash M) + \mu(M \backslash O) \ . \tag{121}$$

Corollary A.3.2. *Let* $(\Omega, \mathfrak{O}, \mu)$ *be a measure space and* $T \in \mathfrak{O}^2$ *be a target then*

$$\mathrm{err}_T^\mu(O) = \mathrm{err}^\mu(T_1, O) \ . \tag{122}$$

The error is thus a general structure of measure spaces and many of the following results are in fact pure measure theory results:

Lemma A.3.1 (Alternative Error Count Definition)**.** *Let* $(\Omega, \mathfrak{O}, \mu)$ *be a measure space then*

$$\forall M, O \in \mathfrak{O} : \mathrm{err}^\mu(M, O) = \mu(M \cup O) - \mu(M \cap O) \tag{123}$$

Proof.

$$\begin{aligned} \mathrm{err}^\mu(M, O) &= \mu(O \cup M) - \mu(O \cap M) \\ &= \mu(\ O \backslash M \ \cup \ M \backslash O \ \cup \ (\ O \cap M)\ ) - \mu(O \cap M) \\ &= \mu(O \backslash M) + \mu(M \backslash O) + \mu(O \cap M) - \mu(O \cap M) \\ &= \mu(O \backslash M) + \mu(M \backslash O) \end{aligned}$$

□

A.4. The Error Metric

Theorem A.4.1 (The Error is a Metric.)**.** *Let* $(\Omega, \mathfrak{O}, \mu)$ *be a measure space then* $(\mathfrak{O}, \mathrm{err}^\mu)$ *is a metric space.*

Proof. Proving symmetry is trivial. The identity of indiscernibles that is

$$\mathrm{err}^\mu(M, O) = 0 \Leftrightarrow O = M$$

has two directions to proof. The $\Leftarrow$ direction is trivial. The other direction implies that

$$\mu(M \cup O) = \mu(M \cap O)$$

and thus

$$\begin{aligned}\mu(O \cup M) &= \mu(\ ((O \cup M) \backslash (O \cap M)) \cup (O \cap M)\ ) \\ &= \mu(\ (O \cup M) \backslash (O \cap M)\ ) + \mu(O \cap M) \\ &= \mu(\ (O \cup M) \backslash (O \cap M)\ ) + \mu(O \cup M)\end{aligned}$$

and so that $0 = \mu(\ (O \cup M) \backslash (O \cap M)\ )$ and thus

$$(O \cup M) \backslash (O \cap M) = \varnothing$$

which then directly implies that

$$(O \cup M) \subseteq (O \cap M)\ .$$

However basic set theory implies that

$$(O \cup M) \supseteq (O \cap M)$$

and thus

$$O \cup M = O \cap M$$

which is false for any set $O \neq M$ and thus

$$O = M\ .$$

The last required property of a metric space the triangle inequality

$$\forall A, B, C \in \mathfrak{O} : \mathrm{err}^{\mu}(A, B) \leqslant \mathrm{err}^{\mu}(A, C) + \mathrm{err}^{\mu}(C, B)$$

is proven in two three steps:

1. For all $X, Y, Z \in \mathfrak{O}$ it holds that

$$\begin{aligned}&\mathrm{err}^{\mu}(X, Y) \\ =&\mu(X \backslash Y) + \mu(Y \backslash X) \\ =&\mu(X \cap Y^{\complement}) + \mu(Y \cap X^{\complement}) \\ =&\mu((X \cap Y^{\complement} \cap Z^{\complement}) \cup (X \cap Y^{\complement} \cap Z)) + \mu((Y \cap X^{\complement} \cap Z^{\complement}) \cup (Y \cap X^{\complement} \cap Z)) \\ =&\mu(X \cap Y^{\complement} \cap Z^{\complement}) + \mu(X \cap Y^{\complement} \cap Z) + \mu(Y \cap X^{\complement} \cap Z^{\complement}) + \mu(Y \cap X^{\complement} \cap Z)\end{aligned}$$

2. Applying the first step this yields

$$\begin{aligned}&\mathrm{err}^{\mu}(A, C) \\ =&\mu(A \cap C^{\complement} \cap B^{\complement}) + \mu(A \cap C^{\complement} \cap B) + \mu(C \cap A^{\complement} \cap B^{\complement}) + \mu(C \cap A^{\complement} \cap B)\end{aligned}$$

and

$$\begin{aligned}&\mathrm{err}^{\mu}(B, C) \\ =&\mu(B \cap C^{\complement} \cap A^{\complement}) + \mu(B \cap C^{\complement} \cap A) + \mu(C \cap B^{\complement} \cap A^{\complement}) + \mu(C \cap B^{\complement} \cap A)\end{aligned}$$

3. Summing the equations of the second step yields

$$\begin{aligned}
&\mathrm{err}^\mu(A,C) + \mathrm{err}^\mu(B,C)\\
=&\mu(A \cap C^\complement \cap B^\complement) + \mu(A \cap C^\complement \cap B) + \mu(C \cap A^\complement \cap B^\complement) + \mu(C \cap A^\complement \cap B)\\
&+ \mu(B \cap C^\complement \cap A^\complement) + \mu(B \cap C^\complement \cap A) + \mu(C \cap B^\complement \cap A^\complement) + \mu(C \cap B^\complement \cap A)\\
\geqslant&\mu(A \cap C^\complement \cap B^\complement) + \mu(C \cap B^\complement \cap A) + \mu(B \cap C^\complement \cap A^\complement) + \mu(C \cap B^\complement \cap A)\\
=&\mu(A \cap B^\complement \cap C^\complement) + \mu(A \cap B^\complement \cap C) + \mu(B \cap A^\complement \cap C^\complement) + \mu(A \cap B^\complement \cap C)\\
=&\mathrm{err}^\mu(A,B)
\end{aligned}$$

which proofs the triangle inequality. □

A.4.1. An Upper Error Bound

Lemma A.4.1 (Upper Error Bound)**.** *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space, then it holds that*

$$\forall A, B \in \mathfrak{O} : \mathrm{err}^\mu(A,B) \leqslant \mu(A) + \mu(B) \tag{124}$$

Proof.

$$\begin{aligned}
\mathrm{err}^\mu(A,B) &= \mu(A \backslash B) + \mu(B \backslash A)\\
&\leqslant \mu(A \backslash B) + \mu(B) + \mu(B \backslash A) + \mu(A)\\
&= \mu(A \backslash B \cup B) + \mu(B \backslash A \cup A)\\
&= \mu(A) + \mu(B)
\end{aligned}$$

□

A.4.2. Defect Value Measure Bounds

Lemma A.4.2 (Defect Value Measure Bounds)**.** *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space, then it holds that*

$$\forall A, B \in \mathfrak{O} : \mu(B) \geqslant \mathrm{err}^\mu(A,B) - \mu(A) \tag{125}$$

$$\forall A, B \in \mathfrak{O} : \mu(B) \geqslant \mu(A) - \mathrm{err}^\mu(A,B) \tag{126}$$

both cases are useful for substitution in proofs and yield the complete bounds equation:

$$\forall A, B \in \mathfrak{O} : |\mu(A) - \mathrm{err}^\mu(A,B)| \leqslant \mu(B) \leqslant |\mu(A) + \mathrm{err}^\mu(A,B)| \tag{127}$$

Proof. The first case is rewriting lemma A.4.1 which yields $\mu(B) \geqslant \mathrm{err}^\mu(A,B) - \mu(A)$

and the second case has the following proof:

$$\begin{aligned}
& \operatorname{err}^{\mu}(A,B) = \mu(A \cup B) - \mu(A \cap B) \\
\Rightarrow\ & \mu(A \cap B) = \mu(A \cup B) - \operatorname{err}^{\mu}(A,B) \\
\Rightarrow\ & \mu(B) \geqslant \mu(A \cup B) - \operatorname{err}^{\mu}(A,B) \\
\Rightarrow\ & \mu(B) + \operatorname{err}^{\mu}(A,B) \geqslant \mu(B \cup A) \\
\Rightarrow\ & \mu(B) + \operatorname{err}^{\mu}(A,B) \geqslant \mu(A) \\
\Rightarrow\ & \mu(B) \geqslant \mu(A) - \operatorname{err}^{\mu}(A,B)
\end{aligned}$$

this also by swapping variables yields

$$\begin{aligned}
& \mu(A) \geqslant \mu(B) - \operatorname{err}^{\mu}(A,B) \\
\Rightarrow\ & \mu(A) + \operatorname{err}^{\mu}(A,B) \geqslant \mu(B) \\
\Rightarrow\ & |\mu(A) + \operatorname{err}^{\mu}(A,B)| \geqslant \mu(B)
\end{aligned}$$

and furthermore

$$\begin{aligned}
& \mu(A) - \operatorname{err}^{\mu}(A,B) \geqslant 0 \\
\Rightarrow\ & |\mu(A) - \operatorname{err}^{\mu}(A,B)| = \operatorname{err}^{\mu}(A,B) - \mu(A) \leqslant \mu(A)
\end{aligned}$$

as well as

$$\begin{aligned}
& \mu(A) - \operatorname{err}^{\mu}(A,B) < 0 \\
\Rightarrow\ & |\mu(A) - \operatorname{err}^{\mu}(A,B)| = |\operatorname{err}^{\mu}(A,B) - \mu(A)| = \operatorname{err}^{\mu}(A,B) - \mu(A) \leqslant \mu(A)
\end{aligned}$$

so that finally

$$|\mu(A) - \operatorname{err}^{\mu}(A,B)| \leqslant \mu(A)$$

□

A.5. Error Factorization

This section provides an theorem useful to calculate actual errors. In practice it is intended for error analysis of systems composed of subsystems, where an function that maps errors to the subsystem-ids of the subsystem they occur in, can be constructed. To discuss how connected subsystems behave under errors relative to each others, it is necessary to be able exactly investigate the "allocation" of errors to the subsystems, which can also be done with the following theorem and appropriate selections of error categorization functions:

Theorem A.5.1 (Error Factorization)**.** *Let* $(\Omega, \mathfrak{O}, \mu)$ *be a measure space,* Y *a countable set and* $f : \Omega \to Y$ *then*

$$\forall A, B \in \mathfrak{O} \ :\ \operatorname{err}^{\mu}(A,B) = \sum_{d \in f(A \cup B)} \operatorname{err}^{\mu}([d]^f_A, [d]^f_B) \tag{128}$$

Proof. Since $[d]_X^f = \varnothing$ for $d \notin f(X)$ for any X we have

$$A = \bigcup A/_f = \bigcup_{d \in f(A)} [d]_A^f = \bigcup_{d \in f(A \cup B)} [d]_A^f$$
$$B = \bigcup B/_f = \bigcup_{d \in f(B)} [d]_B^f = \bigcup_{d \in f(A \cup B)} [d]_B^f$$

with this we have

$$\begin{aligned}
\mu(A \cup B) &= \mu\left(\left(\bigcup_{d \in f(A \cup B)} [d]_A^f\right) \cup \left(\bigcup_{d \in f(A \cup B)} [d]_B^f\right)\right) \\
&= \mu\left(\bigcup_{d \in f(A \cup B)} ([d]_A^f \cup [d]_B^f)\right) \\
&= \sum_{d \in f(A \cup B)} \mu([d]_A^f \cup [d]_B^f)
\end{aligned}$$

and

$$\begin{aligned}
&\mu(A \cap B) \\
=&\mu\left(\left(\bigcup_{d \in f(A \cup B)} [d]_A^f\right) \cap \left(\bigcup_{d' \in f(A \cup B)} [d']_B^f\right)\right) \\
=&\mu\left(\bigcup_{d \in f(A \cup B)} \bigcup_{d' \in f(A \cup B)} ([d]_A^f \cap [d']_B^f)\right) \\
=&\mu\left(\left(\bigcup_{d \in f(A \cup B)} ([d]_A^f \cap [d]_B^f)\right) \cup \left(\bigcup_{d \in f(A \cup B)} \bigcup_{\substack{d' \in f(A \cup B) \\ d' \neq d}} \underbrace{([d]_A^f \cap [d']_B^f)}_{=\varnothing}\right)\right) \\
=&\mu\left(\bigcup_{d \in f(A \cup B)} ([d]_A^f \cap [d]_B^f)\right) \\
=& \sum_{d \in f(A \cup B)} \mu([d]_A^f \cap [d]_B^f)
\end{aligned}$$

and finally

$$\begin{aligned}
\mathrm{err}^{\mu}(A,B) &= \mu(A \cup B) - \mu(A \cap B) \\
&= \left(\sum_{d \in f(A \cup B)} \mu([d]_A^f \cup [d]_B^f) \right) - \left(\sum_{d \in f(A \cup B)} \mu([d]_A^f \cap [d]_B^f) \right) \\
&= \sum_{d \in f(A \cup B)} \left(\mu([d]_A^f \cup [d]_B^f) - \mu([d]_A^f \cap [d]_B^f) \right) \\
&= \sum_{d \in f(A \cup B)} \mathrm{err}^{\mu} \left([d]_A^f, [d]_B^f \right) \ .
\end{aligned}$$

□

The above theorem states that the total number of errors in a system is the sum of errors in its subsystems, provide that an appropriate function to apply the theorem can be constructed. Due to the abundance of emergent behavior of complex systems, this is a deep, since unexpected simple result. It is not investigated if all system designs, that is their specifications into subsystems and how these interact, allow the construction of proper[36] functions as required by the theorem. In that regard there should be even deeper results linking emergent system failures to the existence of proper indicator functions, and thus improper planning and specification[37]. Furthermore there might be an generalization of the result to use functions with non-countable codomains that "replaces "the sums in the expressions with measure integrals.

Corollary A.5.1. *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space, Y a countable set and $f : \Omega \to Y$ then*

$$\forall A, B \in \mathfrak{O} \ : \ \forall d : \mathrm{err}^{\mu} \left(\bigcup A/_f, \bigcup B/_f \right) \geqslant \mathrm{err}^{\mu}([d]_A^f, [d]_B^f) \tag{129}$$

A.6. Principal Counting Outcome Measure Semantics

Due to the lack of an definition of the error at place where the Principal Counting Outcome Measure was defined, it was not possible to show the properties that endows it with the title "Principal". This is done in the following results:

Lemma A.6.1. *The excess measured with the Principal Counting Outcome Measure is the number of observed prohibited results. Formally expressed as: Let $T \in \mathfrak{O}^2$ be a target then*

$$\mu_T^{\#}(\mathrm{excess}_T(O)) = \#(T_0 \cap O) \tag{130}$$

[36] The protocol analysis in the document also indicates that this "proper" is probably related to the input and output of the subsystems.

[37] The author would like to here from anyone who had time to investigate this and got interesting results or has knowledge of already existing interesting results related to that.

Proof.

$$\begin{aligned}
\mathrm{err}_T^{\mu_T^\#}(O) &= \mathrm{err}^{\mu_T^\#}(T_1, O) \\
&= \mu_T^\#(O \backslash T_1) + \mu_T^\#(T_1 \backslash O) \\
&= \#((O \backslash T_1) \cap (T_0 \cup T_1)) + \mu_T^\#(T_1 \backslash O) \\
&= \#((O \cap T_1^\complement) \cap (T_0 \cup T_1)) + \mu_T^\#(T_1 \backslash O) \\
&= \#((O \cap \underbrace{T_1^\complement \cap T_0}_{=T_0}) \cup (O \cap \underbrace{T_1^\complement \cap T_1}_{=\varnothing})) + \mu_T^\#(T_1 \backslash O) \\
&= \#(O \cap T_0) + \mu_T^\#(T_1 \backslash O) \\
&= \#(O \cap T_0) + \mu_T^\#(\mathrm{loss}_T(O)) \\
&= \#(O \cap T_0) + \mathrm{err}_T^{\mu_T^\#}(O) - \mu_T^\#(\mathrm{excess}_T(O))
\end{aligned}$$

and thus $0 = \#(O \cap T_1) - \mu_T^\#(\mathrm{excess}_T(O))$ that is $\mu_T^\#(\mathrm{excess}_T(O)) = \#(O \cap T_1)$. □

Corollary A.6.1. *For the principal outcome measure only prohibited results are relevant excesses.*

Lemma A.6.2. *The loss measured with the Principal Counting Outcome Measure is the number of missed mandatory results. Formally expressed as: Let $T \in \mathfrak{O}^2$ be a target then*

$$\mu_T^\#(\mathrm{loss}_T(O)) = \#(T_1 \cap O^\complement) \tag{131}$$

Proof.

$$\begin{aligned}
\mathrm{err}_T^{\mu_T^\#}(O) &= \mathrm{err}^{\mu_T^\#}(T_1, O) \\
&= \mu_T^\#(O \backslash T_1) + \mu_T^\#(T_1 \backslash O) \\
&= \#((O \backslash T_1) \cap (T_0 \cup T_1)) + \#((T_1 \backslash O) \cap (T_0 \cup T_1)) \\
&= \mu_T^\#(O \backslash T_1) + \#((T_1 \cap O^\complement) \cap (T_0 \cup T_1)) \\
&= \mu_T^\#(O \backslash T_1) + \#((T_1 \cap O^\complement \cap T_0) \cup (T_1 \cap O^\complement \cap T_1)) \\
&= \mu_T^\#(O \backslash T_1) + \#((O^\complement \cap \underbrace{T_0 \cap T_1}_{=\varnothing}) \cup (T_1 \cap O^\complement)) \\
&= \mu_T^\#(O \backslash T_1) + \#(T_1 \cap O^\complement) \\
&= \mu_T^\#(\mathrm{excess}_T(O)) + \#(T_1 \cap O^\complement) \\
&= \mathrm{err}_T^{\mu_T^\#}(O) - \mu_T^\#(\mathrm{loss}_T(O)) + \#(T_1 \cap O^\complement)
\end{aligned}$$

and thus $0 = \#(T_1 \cap O^\complement) - \mu_T^\#(\mathrm{loss}_T(O))$ that is $\mu_T^\#(\mathrm{loss}_T(O)) = \#(T_1 \cap O^\complement)$. □

Corollary A.6.2. *For the Error in the Principal Counting Outcome Measure only prohibited and mandatory results are relevant, since*

$$\mathrm{err}_T^{\mu_T^{\#}}(O) = \#(O \cap T_0) + \#(T_1 \cap O^{\complement}) \ . \tag{132}$$

B. Interpretation Theory

In this section we develop a theory on how reconstruct an observable result that was reported by different sources to allow observation data recovery in case of data loss, corruption or fraud. In fact it provides a general theory on how to interpret inconsistent source data as consistent as possible.

B.1. Sourced Key Value Structure

The basic data of this discussion must have three components, a source of some information, a key referring to some information and the information itself. This is captured by the following base definition:

Definition B.1.1 (Sourced Key Value Structure)**.** *A tuple*

$$(\Omega, K, S, V, f_{\text{key}}, f_{\text{val}}, f_{\text{src}}, v)$$

where Ω is finite[38] set and the type base data of the structure, K is the type of the keys used, S is the type of the sources used, V is the type of values used, $f_{\text{key}} : \Omega \to K$ provides the key of a datum, $f_{\text{src}} : \Omega \to S$ provides the source of a datum, $f_{\text{val}} : \Omega \to V$ provides the value of a datum, and $v \in \mathbb{N}^+$ is a significance threshold, called the byzantine failure threshold[39] due to results shown later, must satisfy

$$\#S \geqslant 2v - 1 \tag{133}$$

.

The above structure does not contain any data only data types, and thus can be used to investigate any actual collection of data for which a sourced key value structure can be defined. Furthermore the results derived can be understood as part of formal semiotics as will be shortly discussed later[40]. For the semiotic context the keys are the signs, the values are the signifieds and the sources are a subset of all of the interpreters. However using this terminology does not yield a nice named functional interface[41] which will defined next since the above structure is a bit unwieldy:

[38] The finiteness condition can probably be relaxed, but actual infinite sets are not requited for the protocol analysis and would certainly add formal clutter to some of the discussions below.

[39] That is the lowest number of errors at which a byzantine failure can not be precluded as will be shown as terminal result of this theory. Thus 0 is not a valid value, since it would allow for byzantine failures in the absence of errors.

[40] This derivation of results for formal semiotics happened by accident and was noticed by the author while cleaning up the results of the protocol analysis for presentation to third parties. However despite being surprising, at least for the author, it has deep implications for security research as will be discussed in the non formal sections at the end of this appendix.

[41] This means that there are too many words with the same initial letters, slowing down reading and decoding speed of presented content, be it formal or not, in that nomenclature.

B.1.1. Data

Definition B.1.2. *Let* $\mathcal{V} = (\Omega, K, S, V, f_{\text{key}}, f_{\text{val}}, f_{\text{src}}, v)$ *be a sourced key value structure then*

$$\Omega_{\mathcal{V}} := \Omega \tag{134}$$

B.1.2. Key

Definition B.1.3. *Let* $\mathcal{V} = (\Omega, K, S, V, f_{\text{key}}, f_{\text{val}}, f_{\text{src}}, v)$ *be a sourced key value structure then*

$$\begin{aligned} \text{key}_{\mathcal{V}} &: \Omega \to K \\ \text{key}_{\mathcal{V}} &:= f_{\text{key}} \end{aligned} \tag{135}$$

B.1.3. Value

Definition B.1.4. *Let* $\mathcal{V} = (\Omega, K, S, V, f_{\text{key}}, f_{\text{val}}, f_{\text{src}}, v)$ *be a sourced key value structure then*

$$\begin{aligned} \text{val}_{\mathcal{V}} &: \Omega \to V \\ \text{val}_{\mathcal{V}} &:= f_{\text{val}} \end{aligned} \tag{136}$$

B.1.4. Source

Sources identify the source of datum copy.

Definition B.1.5. *Let* $\mathcal{V} = (\Omega, K, S, V, f_{\text{key}}, f_{\text{val}}, f_{\text{src}}, v)$ *be a sourced key value structure then*

$$\begin{aligned} \text{src}_{\mathcal{V}} &: \Omega \to S \\ \text{src}_{\mathcal{V}} &:= f_{\text{src}} \end{aligned} \tag{137}$$

B.1.5. Byzantine Failure Threshold

Definition B.1.6. *Let* $\mathcal{V} = (\Omega, K, S, V, f_{\text{key}}, f_{\text{val}}, f_{\text{src}}, v)$ *be a sourced key value structure then*

$$\begin{aligned} \text{bft}_{\mathcal{V}} &\in \mathbb{N} \\ \text{bft}_{\mathcal{V}} &:= v \end{aligned} \tag{138}$$

B.2. Associations

Associations can be understood as a formal representation of how a particular source links a key of to a value.

B.2.1. Definition

Definition B.2.1 (Association Reference)**.** *Let $\mathcal{V}$ be a sourced key value structure then*

$$\begin{aligned} &\mathrm{ar}_{\mathcal{V}} : \Omega_{\mathcal{V}} \to \mathrm{codom}(\mathrm{src}_{\mathcal{V}}) \times \mathrm{codom}(\mathrm{key}_{\mathcal{V}}) \\ &\mathrm{ar}_{\mathcal{V}} := d \mapsto (\mathrm{src}_{\mathcal{V}}(d), \mathrm{key}_{\mathcal{V}}(d)) \end{aligned} \tag{139}$$

Definition B.2.2 (Association)**.** *Let $\mathcal{V}$ be a sourced key value structure $r \in \mathrm{codom}(\mathrm{ar}_{\mathcal{V}})$ be a association reference, then the association derived from $\mathcal{D} \subseteq \Omega_{\mathcal{V}}$ is defined as*

$$[r]_{\mathcal{D}}^{\mathrm{ar}_{\mathcal{V}}} \in 2^{\Omega_{\mathcal{V}}} \tag{140}$$

where no new syntax or name is defined, since the above preimage notation is compact and readably already

Definition B.2.3 (Set of Derived Associations)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the set of associations derived from $\mathcal{D} \subseteq \Omega_{\mathcal{V}}$ is*

$$\mathcal{D}/_{\mathrm{ar}_{\mathcal{V}}} \tag{141}$$

where no new syntax or name is defined, since the above equivalence class syntax is short and precise.

Definition B.2.4 (The Associations Type)**.** *Let $\mathcal{V}$ be a sourced key value structure then its associated set of all possible associations is*

$$\mathfrak{A}_{\mathcal{V}} := \bigcup_{\mathcal{D} \in 2^{\Omega_{\mathcal{V}}}} \mathcal{D}/_{\mathrm{ar}_{\mathcal{V}}} \tag{142}$$

where no new syntax or name is defined, since the above preimage notation is compact and readably already

Corollary B.2.1.

$$\mathfrak{A}_{\mathcal{V}} \subseteq 2^{\Omega_{\mathcal{V}}} \tag{143}$$

Definition B.2.5 (Association Key)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the association key is the*

$$\begin{aligned} &\mathrm{ak}_{\mathcal{V}} : \mathfrak{A}_{\mathcal{V}} \to \mathrm{codom}(\mathrm{key}_{\mathcal{V}}) \\ &\mathrm{ak}_{\mathcal{V}} := A \mapsto \Big\{ k \text{ if } \{k\} = \mathrm{key}_{\mathcal{V}}(A) \end{aligned} \tag{144}$$

that is the unpacked singleton of datum keys.

Corollary B.2.2. *The association key is well defined, i.e.* $\mathrm{key}_{\mathcal{V}}(A)$ *is always a singleton.*

The following definition is a formality to fix the domain and formally apply the pointwise application syntax to sets of associations:

Definition B.2.6 (Associated Values)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the associated values are*

$$\begin{aligned} &\mathrm{av}_{\mathcal{V}} : \mathfrak{A}_{\mathcal{V}} \to 2^{\mathrm{codom}(\mathrm{val}_{\mathcal{V}})} \\ &\mathrm{av}_{\mathcal{V}} := A \mapsto \mathrm{val}_{\mathcal{V}}(A) \end{aligned} \tag{145}$$

that is the point wise application of $\mathrm{val}_{\mathcal{V}}$ *to the data of the associations.*

B.2.2. Definite

Definition B.2.7 (Definite Association)**.** *Let $\mathcal{V}$ be a sourced key value structure and $A \in \mathfrak{A}$ then A definite by definition if*

$$\#\mathrm{val}_{\mathcal{V}}(A) = 1 \ . \tag{146}$$

This definition also is the explanation for the necessity of defining associations. On design requirement for this theory is the ability to analyze byzantine failure tolerance. This implies existence of malicious sources. A single source could create arbitrary many different values for the same key and raise any associated basic error for associations towards infinity. However it can by definition only damage its own associations, for any given association reference. Even more generally anyone can only create damage to associations of source he is able to appear as. This in practices links errors counted in associations to compromised digital signatures or identifiable openly hostile signers.

B.3. Significations

Significantions formalize the relation between keys and values[42] for sets of sources.

B.3.1. Definition

Definition B.3.1 (Signification Relation)**.** *Let $\mathcal{V}$ be a sourced key value structure then*

$$\begin{aligned} &\mathrm{sr}_{\mathcal{V}} : \mathfrak{A}_{\mathcal{V}} \to \mathrm{codom}(\mathrm{key}_{\mathcal{V}}) \times 2^{\mathrm{codom}(\mathrm{val}_{\mathcal{V}})} \\ &\mathrm{sr}_{\mathcal{V}} := A \mapsto (\mathrm{ak}_{\mathcal{V}}(A), \mathrm{av}_{\mathcal{V}}(A)) \end{aligned} \tag{147}$$

Definition B.3.2 (Signification)**.** *Let $\mathcal{V}$ be a sourced key value structure and $r \in \mathrm{codom}(\mathrm{ar}_{\mathcal{V}})$ be a signification relation, then the signification of r created from $\mathcal{A} \subseteq \mathfrak{A}_{\mathcal{V}}$ is defined as*

$$[r]^{\mathrm{sr}_{\mathcal{V}}}_{\mathcal{A}} \in 2^{\mathfrak{A}_{\mathcal{V}}} \tag{148}$$

where no new syntax or name is defined.

Definition B.3.3 (The Significations Type)**.** *Let $\mathcal{V}$ be a sourced key value structure then its associated set of all possible significantions is*

$$\mathfrak{S}_{\mathcal{V}} := \bigcup_{\mathcal{A} \in 2^{\mathfrak{A}_{\mathcal{V}}}} \mathcal{A}/_{\mathrm{sr}_{\mathcal{V}}} \ . \tag{149}$$

Corollary B.3.1.

$$\mathfrak{S}_{\mathcal{V}} \subseteq 2^{\mathfrak{A}_{\mathcal{V}}} \subseteq 2^{2^{\Omega_{\mathcal{V}}}} \tag{150}$$

[42] Here it would really lead to confusion calling keys signifiers, values signifieds, and the relation between them Significantions.

Definition B.3.4 (Signification Key)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the signification key is the*

$$\begin{aligned} &\mathrm{sk}_{\mathcal{V}} : \mathfrak{S}_{\mathcal{V}} \to \mathrm{codom}(\mathrm{key}_{\mathcal{V}}) \\ &\mathrm{sk}_{\mathcal{V}} := S \mapsto \left\{ k \text{ if } \{k\} = \mathrm{ak}_{\mathcal{V}}(S) \right. \end{aligned} \tag{151}$$

that is the unpacked singleton of association keys.

Definition B.3.5 (Signified Values)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the indicated values are*

$$\begin{aligned} &\mathrm{sv}_{\mathcal{V}} : \mathfrak{S}_{\mathcal{V}} \to 2^{\mathrm{codom}(\mathrm{val}_{\mathcal{V}})} \\ &\mathrm{sv}_{\mathcal{V}} := S \mapsto \left\{ V \text{ if } \{V\} = \mathrm{av}_{\mathcal{V}}(S) \right. \end{aligned} \tag{152}$$

that is the unpacked singleton of associated values.

Corollary B.3.2. *If a single association of an signification is definite, all associations of the signification are definite, i.e .*

$$\forall S \in \mathfrak{S}_{\mathcal{V}} : ((\exists A \in S : \#\mathrm{av}_{\mathcal{V}}(A) = 1) \Rightarrow (\forall A \in S : \#\mathrm{av}_{\mathcal{V}}(A) = 1)) \tag{153}$$

Proof. The signification is by definition an equivalence class of associations having the same reference and thus by construction the same values. □

Corollary B.3.3. *If a single association of an signification is indefinite, all associations of the signification are indefinite.*

B.3.2. Significance

The following measure for the significance simply counts the definite associations of an signification:

Definition B.3.6 (Significance Measure)**.** *The measure*

$$\begin{aligned} &\mu_{\mathcal{V}} : 2^{\mathfrak{A}_{\mathcal{V}}} \to \mathbb{N} \\ &\mu_{\mathcal{V}} := \mathcal{A} \mapsto \#(\mathcal{A} \cap \{A \in \mathfrak{A}_{\mathcal{V}} \mid \#\mathrm{av}_{\mathcal{V}}(A) = 1\}) \end{aligned} \tag{154}$$

of the measure space $(\mathfrak{A}_{\mathcal{V}}, 2^{\mathfrak{A}_{\mathcal{V}}}, \mu_{\mathcal{V}})$ is called significance measure and its values are called significance.

Lemma B.3.1. *The significance measure is indeed a measure of a measure space.*

Proof. The above measure is formally the principal counting outcome measure for the target

$$(\varnothing, \{A \in \mathfrak{A}_{\mathcal{V}} \mid \#\mathrm{val}_{\mathcal{V}}(U) = 1\})$$

also this target is a plain formal construction and probably of no interest in most contexts. Further the power set is allays a σ-algebra called the discrete σ-algebra. □

Corollary B.3.4. *Only significantions of definite associations are relevant to the significance measure.*

The following lemma explains why the domain of the significance measure is not the type of significations:

Lemma B.3.2. *In general $\mathfrak{S}_\mathcal{V}$ is not a σ-Algebra.*

Proof. There are may conunter examples that lack closedness under complements. The existence proof is a corollary. □

B.3.3. Definite

Definition B.3.7 (Definite Signification)**.** *Let $\mathcal{V}$ be a sourced key value structure then $I \in \mathfrak{S}_\mathcal{V}$ is definite by definition if*

$$\#\mathrm{sv}_\mathcal{V}(S) = 1 \ . \tag{155}$$

Lemma B.3.3 (Relevant Significantions are Definite)**.**

$$\forall S \in \mathfrak{S}_\mathcal{V} : \mu_\mathcal{V}(S) > 0 \Rightarrow \#\mathrm{sv}(S) = 1 \tag{156}$$

Proof.

$$\begin{aligned}
& \mu_\mathcal{V}(S) > 0 \\
\Rightarrow\ & \#(S \cap \{A \in \mathfrak{A}_\mathcal{V} \mid \#\mathrm{av}_\mathcal{V}(A) = 1\}) > 0 \\
\Rightarrow\ & \exists A \in S : \#\mathrm{av}_\mathcal{V}(A) = 1 \\
\Rightarrow\ & \forall A \in S : \#\mathrm{av}_\mathcal{V}(A) = 1 && (\text{ By corollary B.3.2 }) \\
\Rightarrow\ & \exists V \in \mathrm{codom}(\mathrm{val}_\mathcal{V}) : \forall A \in S : \mathrm{av}_\mathcal{V}(A) = \{V\} \\
\Rightarrow\ & \exists V \in \mathrm{codom}(\mathrm{val}_\mathcal{V}) : \forall A \in S : \mathrm{av}_\mathcal{V}(A) = \mathrm{sv}(S) && (\text{ By Definition B.3.5 }) \\
\Rightarrow\ & \exists V \in \mathrm{codom}(\mathrm{val}_\mathcal{V}) : \forall A \in S : 1 = \#\mathrm{av}_\mathcal{V}(A) = \#\mathrm{sv}(S) \\
\Rightarrow\ & 1 = \#\mathrm{sv}(S)
\end{aligned}$$

□

B.4. Formations

B.4.1. Definition

If an interpreter is given a key and ought deduce the value the key refers to then he/she/it must evaluate the significations associated with that key. Formations are the sets of significantions of the same key used for such evaluations:

Definition B.4.1 (Formation)**.** *Let $\mathcal{V}$ be a sourced key value structure and $k \in \mathrm{codom}(\mathrm{key}_\mathcal{V})$ be a key, then the formation of k created from $S \subseteq \mathfrak{S}_\mathcal{V}$ is defined as*

$$[k]_S^{\mathrm{sk}_\mathcal{V}} \in 2^{\mathfrak{S}_\mathcal{V}} \tag{157}$$

where no new syntax or name is defined.

Definition B.4.2 (The Formations Type)**.** *Let $\mathcal{V}$ be a sourced key value structure then its associated set of all possible formation is*

$$\mathfrak{F}_{\mathcal{V}} := \bigcup_{\mathcal{S} \in 2^{\mathfrak{S}_{\mathcal{V}}}} \mathcal{S}/_{\mathrm{sk}_{\mathcal{V}}} \tag{158}$$

Corollary B.4.1.

$$\mathfrak{F}_{\mathcal{V}} \subseteq 2^{\mathfrak{S}_{\mathcal{V}}} \subseteq 2^{2^{\mathfrak{A}_{\mathcal{V}}}} \subseteq 2^{2^{2^{\Omega_{\mathcal{V}}}}} \tag{159}$$

Definition B.4.3 (Formation Key)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the formation key is the*

$$\begin{aligned} &\mathrm{fk}_{\mathcal{V}} : \mathfrak{F}_{\mathcal{V}} \to \mathrm{codom}(\mathrm{key}_{\mathcal{V}}) \\ &\mathrm{fk}_{\mathcal{V}} := F \mapsto \Big\{ k \text{ if } \{k\} = \mathrm{sk}_{\mathcal{V}}(F) \end{aligned} \tag{160}$$

that is the unpacked singleton of signification keys.

Different sources might "provide" different values for a key, so to readably access the values entailed by a formation the following definition is required:

Definition B.4.4 (Formation Values)**.** *Let $\mathcal{V}$ be a sourced key value structure, then the formation values are*

$$\begin{aligned} &\mathrm{fv}_{\mathcal{V}} : \mathfrak{F}_{\mathcal{V}} \to 2^{\mathrm{codom}(\mathrm{val}_{\mathcal{V}})} \\ &\mathrm{fv}_{\mathcal{V}} := F \mapsto \bigcup_{S \in F} \mathrm{sv}_{\mathcal{V}}(S) \end{aligned} \tag{161}$$

that is the unpacked singleton of signification keys.

B.4.2. Definite

Definition B.4.5 (Definite Formation)**.** *Let $\mathcal{V}$ be a sourced key value structure then $F \in \mathfrak{F}_{\mathcal{V}}$ is definite by definition if*

$$\#\mathrm{fv}_{\mathcal{V}}(F) = 1 \ . \tag{162}$$

B.4.3. Cleared

Definition B.4.6 (Cleared Situation[43])**.** *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space*

$$\begin{aligned} &\mathrm{clrd}^{\mu}_{v} : 2^{\mathfrak{O}} \to 2^{\mathfrak{O}} \\ &\mathrm{clrd}^{\mu}_{v} := \mathcal{F} \mapsto \{S \in \mathcal{F} \mid \mu(S) \geqslant v\} \end{aligned} \tag{163}$$

where $v \in \mathbb{N}$ is called a relevancy threshold.

[43] A situation is a set of outcomes in the context of the above error theory. So this definition and some of the following results might be cleaned up later and moved to the error theory appendix. Also using "cleared" is a wordplay for mnemonic support, indicating that something has become more clear and having eliminated threats to meaning.

Definition B.4.7 (Cleared Formation)**.** *Let $\mathcal{V}$ be a sourced key value structure and $F \subseteq \mathfrak{O}$ then*

$$\begin{aligned} &\mathrm{cf}_{\mathcal{V}} : \mathfrak{F}\mathcal{V} \to \mathfrak{F}\mathcal{V} \\ &\mathrm{cf}_{\mathcal{V}} := F \mapsto \mathrm{clrd}^{\mu_{\mathcal{V}}}_{\mathrm{bft}_{\mathcal{V}}}(F) \end{aligned} \tag{164}$$

Lemma B.4.1. *The cleared formation is well defined, that is: Let $\mathcal{V}$ be a sourced key value structure, then*

$$\forall F \in \mathfrak{F}\mathcal{V} : \mathrm{clrd}^{\mu_{\mathcal{V}}}_{\mathrm{bft}_{\mathcal{V}}}(F) \in \mathfrak{F}\mathcal{V} \tag{165}$$

Proof. Let

$$F^* := \{S \in F \,|\, \mu(S) \geqslant v \wedge \mathrm{sk}_{\mathcal{V}}(S) = \mathrm{fk}_{\mathcal{V}}(F)\}$$

then $F^* \subseteq F \subseteq \mathfrak{A}_{\mathcal{V}}$ corollary B.4.1 and construction and thus $F^* \in 2^{\mathfrak{A}_{\mathcal{V}}}$. Next we note that $F^*/_{\mathrm{ak}_{\mathcal{V}}} = \{F^*\}$ and by definition B.4.2 it follows that $F^* \in \mathfrak{F}\mathcal{V}$. Last we recognize that

$$\begin{aligned} \mathrm{clrd}^{\mu_{\mathcal{V}}}_{\mathrm{bft}_{\mathcal{V}}}(F) &= \{S \in F \mid \mu(S) \geqslant v\} \\ &= \{S \in F \mid \mu(S) \geqslant v \wedge \mathrm{sk}_{\mathcal{V}}(S) = \mathrm{fk}_{\mathcal{V}}(F)\} \\ &= F^* \end{aligned}$$

□

Corollary B.4.2. *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space then it holds for all $\mathcal{F} \in 2^{\mathfrak{O}}$ that*

$$\mathrm{clrd}^{\mu}_{0}(\mathcal{F}) = \mathcal{F} \tag{166}$$

$$\forall v \in \mathbb{N} : v > \sup(\mu(\mathcal{F})) \Rightarrow \mathrm{clrd}^{\mu}_{v}(\mathcal{F}) = \varnothing \tag{167}$$

Lemma B.4.2. *Let $(\Omega, \mathfrak{O}, \mu)$ be a measure space then it holds for all $\mathcal{F} \in 2^{\mathfrak{O}}$ that*

$$\forall v, v' \in \mathbb{N} : v \leqslant v' \;\Rightarrow \mathrm{clrd}^{\mu}_{v}(\mathcal{F}) \supseteq \mathrm{clrd}^{\mu}_{v'}(\mathcal{F}) \tag{168}$$

Proof. For all $\forall v, v' \in \mathbb{N}$ with $v \leqslant v'$ it follows

$$\begin{aligned} \mathrm{clrd}^{\mu}_{v'}(\mathcal{F}) &= \left\{\; S \in \mathcal{F} \;\middle|\; \mu(S) \geqslant v' \;\right\} \\ &= \left\{\; S \in \mathcal{F} \;\middle|\; \mu(S) \geqslant v' \wedge \mu(S) \geqslant v \;\right\} \\ &\subseteq \{\; S \in \mathcal{F} \mid \mu(S) \geqslant v \;\} \\ &= \mathrm{clrd}^{\mu}_{v}(\mathcal{F}) \end{aligned}$$

□

Corollary B.4.3. *Let $(\Omega, \mathfrak{O}, \mu)$be a measure space then it holds for all $\mathcal{F} \in 2^{\mathfrak{O}}$ that*

$$\forall v, v' \in \mathbb{N} : v \leqslant v' \;\Rightarrow \#\mathrm{clrd}^{\mu}_{v}(\mathcal{F}) \geqslant \#\mathrm{clrd}^{\mu}_{v'}(\mathcal{F}) \tag{169}$$

Lemma B.4.3. *All Significations in a cleared formation are definite, formally*

$$\mathrm{cf}_{\mathcal{V}}(F) \neq \varnothing \Rightarrow \forall S \in F : \#\mathrm{sv}(S) = 1 \tag{170}$$

Proof. By definition $\mathrm{bft}_{\mathcal{V}} \geqslant 1$, but non definite formations have signification level 0, so that they get filtered out by $\mathrm{clrd}^{\mu_{\mathcal{V}}}_{\mathrm{bft}_{\mathcal{V}}}$. □

B.4.4. Condition

The condition of a formation is the maximal significance for which there is at least one signification:

Definition B.4.8 (Condition)**.** *Let $\mathcal{V}$ be a sourced key value structure than the condition is defined as*

$$\begin{aligned} &\mathrm{cnd}_{\mathcal{V}} : \mathfrak{F}_{\mathcal{V}} \to \mathbb{N} \\ &\mathrm{cnd}_{\mathcal{V}} := F \mapsto \max\left(\ \{ v \in \mathbb{N} \,|\, \#\mathrm{clrd}_v^{\mu_{\mathcal{V}}}(F) \geqslant 1\} \cup \{0\} \ \right) \end{aligned} \tag{171}$$

Lemma B.4.4. *Let $\mathcal{V}$ be a sourced key value structure then*

$$\forall v \in \mathbb{N} : \forall F \in \mathfrak{F}_{\mathcal{V}} : 0 < v \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \Rightarrow \#\mathrm{clrd}_v^{\mu_{\mathcal{V}}}(F) \geqslant 1 \tag{172}$$

Proof.

$$\begin{aligned} & 0 < v \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \\ \Rightarrow\ & 0 < v \leqslant \max\left(\left\{v' \in \mathbb{N} \middle| \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1\right\} \cup \{0\}\right) \\ \Rightarrow\ & 0 < v \leqslant \max\left(\left\{v' \in \mathbb{N} \middle| \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \vee v' = 0\right\}\right) \\ \Rightarrow\ & \exists v' \in \left\{v' \in \mathbb{N} \middle| \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \vee v' = 0\right\} : 0 < v \leqslant v' \\ \Rightarrow\ & \exists v' \in \mathbb{N} : (\#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \vee v' = 0) \wedge (0 < v \leqslant v') \\ \Rightarrow\ & \exists v' \in \mathbb{N} : (\#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \vee v' = 0) \wedge (0 \neq v') \wedge (v \leqslant v') \\ \Rightarrow\ & \exists v' \in \mathbb{N} : (\#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \wedge (0 \neq v') \wedge (v \leqslant v')) \vee ((v' = 0) \wedge (0 \neq v') \wedge (v \leqslant v')) \\ \Rightarrow\ & \exists v' \in \mathbb{N} : \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \wedge (0 \neq v') \wedge (v \leqslant v') \\ \Rightarrow\ & \exists v' \in \mathbb{N} : \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \geqslant 1 \wedge (v \leqslant v') \\ \Rightarrow\ & \#\mathrm{clrd}_v^{\mu_{\mathcal{V}}}(F) \geqslant 1 \qquad \text{(By corollary B.4.3)} \end{aligned}$$

□

Lemma B.4.5. *Let $\mathcal{V}$ be a sourced key value structure then*

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : \forall S \in F : \mu_{\mathcal{V}}(S) \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \tag{173}$$

Proof.

$$\begin{aligned}
S \in F \Rightarrow\ \mu_{\mathcal{V}}(S) &\leqslant \max(\mu_{\mathcal{V}}(F)) \\
&= \max(\{\, \mu_{\mathcal{V}}(S') \,|\, S' \in F \,\}) \\
&= \max(\{\, s \in \mathbb{N} \,|\, \exists S' \in F : \mu_{\mathcal{V}}(S') = s \,\}) \\
&\leqslant \max(\{\, s \in \mathbb{N} \,|\, \exists S' \in F : \mu_{\mathcal{V}}(S') \geqslant s \,\}) \\
&= \max(\{\, s \in \mathbb{N} \,|\, \exists S' \in F : \mu_{\mathcal{V}}(S') \geqslant s \wedge S' \in \mathrm{clrd}_s^{\mu_{\mathcal{V}}}(F) \,\}) \\
&= \max(\{\, s \in \mathbb{N} \,|\, \exists S' \in F : S' \in \mathrm{clrd}_s^{\mu_{\mathcal{V}}}(F) \,\}) \\
&= \max(\{\, s \in \mathbb{N} \,|\, \#\mathrm{clrd}_s^{\mu_{\mathcal{V}}}(F) \geqslant 1 \,\}) \\
&= \max(\{\, s \in \mathbb{N} \,|\, \#\mathrm{clrd}_s^{\mu_{\mathcal{V}}}(F) \geqslant 1 \,\} \cup \{0\}) \\
&= \mathrm{cnd}_{\mathcal{V}}(F)
\end{aligned}$$

□

B.4.5. Corruption

The corruption of a formation is the significance for which there are indefinite, i.e. ambiguous, significations:

Definition B.4.9 (Formation Corruption)**.** *Let $\mathcal{V}$ be a sourced key value structure then*[44]

$$\begin{aligned}
&\mathrm{crr}_{\mathcal{V}} : \mathfrak{F}_{\mathcal{V}} \to \mathbb{N} \\
&\mathrm{crr}_{\mathcal{V}} := F \mapsto \max(\ \ \{v \in \mathbb{N} \,|\, \#\mathrm{clrd}_v^{\mu_{\mathcal{V}}}(F) > 1\} \cup \{0\}\ \ )
\end{aligned} \tag{174}$$

Corollary B.4.4. *Let $\mathcal{V}$ be a sourced key value structure then it holds that*

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : \mathrm{crr}_{\mathcal{V}}(F) \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \tag{175}$$

Lemma B.4.6.

$$\mathrm{crr}_{\mathcal{V}}(F) < v \Rightarrow \#\mathrm{clrd}_v^{\mu_{\mathcal{V}}}(F) \leqslant 1 \tag{176}$$

Proof.

$$\begin{aligned}
&\mathrm{crr}_{\mathcal{V}}(F) < v \\
\Rightarrow\ &\max\left(\left\{v' \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) > 1\right\} \cup \{0\}\right) < v \\
\Rightarrow\ &\forall v' \in \left\{v' \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) > 1\right\} \cup \{0\} : v' < v \\
\Rightarrow\ &\forall v' \in \left\{v' \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) > 1 \vee v' = 0\right\} : v' < v \\
\Rightarrow\ &v \notin \left\{v' \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) > 1 \vee v' = 0\right\} \\
\Rightarrow\ &v \in \left\{v' \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \leqslant 1 \wedge v \neq 0\right\} \\
\Rightarrow\ &v \in \left\{v' \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \leqslant 1\right\} \\
\Rightarrow\ &\#\mathrm{clrd}_{v'}^{\mu_{\mathcal{V}}}(F) \leqslant 1
\end{aligned}$$

□

[44] The name "corr" or "cor" might seem closer, but is usually used for correlation.

Lemma B.4.7. *Let $\mathcal{V}$ be a sourced key value structure then it holds that*

$$\forall v \in \mathbb{N} : \forall F \in \mathfrak{F}_{\mathcal{V}} : \mathrm{crr}_{\mathcal{V}}(F) < v \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \Rightarrow \#\mathrm{clrd}_v^{\mu\nu}(F) = 1 \tag{177}$$

Proof. Lemma B.4.6 directly implies $\#\mathrm{clrd}_v^{\mu\nu}(F) \leqslant 1$. From the definition of $\mathrm{crr}_{\mathcal{V}}$ it follows that $0 \leqslant \mathrm{crr}_{\mathcal{V}}$ and thus $0 < v \leqslant \mathrm{cnd}^{\mu}(F)$ yielding $\#\mathrm{clrd}_v^{\mu\nu}(F) \geqslant 1$ by lemma B.4.4 and therefore $\#\mathrm{clrd}_v^{\mu\nu}(A) = 1$. □

Corollary B.4.5. *Let $\mathcal{V}$ be a sourced key value structure then it holds that*

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : \mathrm{crr}_{\mathcal{V}}(F) < \mathrm{bft}_{\mathcal{V}} \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \Rightarrow \#\mathrm{cf}_{\mathcal{V}}(F) = 1 \tag{178}$$

B.4.6. Integrity

Definition B.4.10 (Formation Integrity)**.** *Let $\mathcal{V}$ be a sourced key value structure then the formation integrity is defined as:*

$$\mathrm{fi}_{\mathcal{V}}(F) := \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F)\,,\, \mathrm{cnd}_{\mathcal{V}}(F) + 1 - \mathrm{bft}_{\mathcal{V}}\,) \tag{179}$$

Lemma B.4.8. *Let $\mathcal{V}$ be a sourced key value structure, then*

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : 0 < \mathrm{fi}_{\mathcal{V}}(F) \Rightarrow 0 \leqslant \mathrm{crr}_{\mathcal{V}}(F) < \mathrm{bft}_{\mathcal{V}} \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \tag{180}$$

Proof. First note that

$$\begin{aligned} & 0 < \mathrm{fi}_{\mathcal{V}}(F) = \min(\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F), \mathrm{cnd}_{\mathcal{V}}(F) + 1 - \mathrm{bft}_{\mathcal{V}}) \leqslant \mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F) \\ \Rightarrow\ & 0 < \mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F) \\ \Rightarrow\ & \mathrm{crr}_{\mathcal{V}}(F) < \mathrm{bft}_{\mathcal{V}} \end{aligned}$$

next quite analog

$$\begin{aligned} & 0 < \mathrm{di}^{v}(F) = \min(\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F), \mathrm{cnd}_{\mathcal{V}}(F) + 1 - \mathrm{bft}_{\mathcal{V}}) \leqslant \mathrm{cnd}(F) + 1 - \mathrm{bft}_{\mathcal{V}} \\ \Rightarrow\ & 0 < \mathrm{cnd}_{\mathcal{V}}(F) + 1 - \mathrm{bft}_{\mathcal{V}} \\ \Rightarrow\ & \mathrm{bft}_{\mathcal{V}} < \mathrm{cnd}_{\mathcal{V}}(F) + 1 \\ \Rightarrow\ & \mathrm{bft}_{\mathcal{V}} \leqslant \mathrm{cnd}_{\mathcal{V}}(F) \end{aligned}$$

and thus $\mathrm{crr}_{\mathcal{V}}(F) < \mathrm{bft}_{\mathcal{V}} \leqslant \mathrm{cnd}_{\mathcal{V}}(F)$. Finally the by definition of $\mathrm{crr}_{\mathcal{V}}(F)$ it holds that $0 \leqslant \mathrm{crr}_{\mathcal{V}}(F)$. □

Corollary B.4.6. *Every integer formation is definite, or formally: Let $\mathcal{V}$ be a sourced key value structure, then it holds that*

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : 0 < \mathrm{fi}_{\mathcal{V}}(F) \Rightarrow \#\mathrm{cf}_{\mathcal{V}}(F) = 1\ . \tag{181}$$

B.4.7. Intact

Definition B.4.11 (Intact Formations)**.** *Let $\mathcal{V}$ be a sourced key value structure then $F \in \mathfrak{F}_{\mathcal{V}}$ is intact if*

$$\mathrm{crr}_{\mathcal{V}}(F) = 0 \wedge \mathrm{cnd}_{\mathcal{V}}(F) = \#\mathrm{dom}(\mathrm{src}_{\mathcal{V}}) \tag{182}$$

Corollary B.4.7. *If all sources conspire intact formations can lead to valid unsound interpretations.*

Lemma B.4.9 (Intact Formation Integrity)**.** *Let $\mathcal{V}$ be an sourced key value structure and $F^* \in \mathfrak{F}_{\mathcal{V}}$ be a intact formation then it holds that*

$$\mathrm{fi}_{\mathcal{V}}(F^*) = \mathrm{bft}_{\mathcal{V}} \ . \tag{183}$$

Proof. The definition of the sourced key value structure, in particular eqn. 133 yields

$$\begin{aligned}
& \#\mathrm{dom}(\mathrm{src}_{\mathcal{V}}) \geqslant 2\,\mathrm{bft}_{\mathcal{V}} - 1 \\
\Rightarrow\ & \#\mathrm{dom}(\mathrm{src}_{\mathcal{V}}) + 1 - \mathrm{bft}_{\mathcal{V}} \geqslant \mathrm{bft}_{\mathcal{V}} \\
\Rightarrow\ & \min(\,\mathrm{bft}_{\mathcal{V}}\,,\, \#\mathrm{dom}(\mathrm{src}_{\mathcal{V}}) + 1 - \mathrm{bft}_{\mathcal{V}}) = \mathrm{bft}_{\mathcal{V}} \\
\Rightarrow\ & \min(\,\mathrm{bft}_{\mathcal{V}}\,,\, \mathrm{cnd}_{\mathcal{V}}(F^*) + 1 - \mathrm{bft}_{\mathcal{V}}\,) = \mathrm{bft}_{\mathcal{V}} \\
\Rightarrow\ & \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F^*)\,,\, \mathrm{cnd}_{\mathcal{V}}(F^*) + 1 - \mathrm{bft}_{\mathcal{V}}\,) = \mathrm{bft}_{\mathcal{V}} \\
\Rightarrow\ & \mathrm{fi}_{\mathcal{V}}(F^*) = \mathrm{bft}_{\mathcal{V}}
\end{aligned}$$

□

B.4.8. Defect

In this section the relation between the association error and the integrity level is thus further investigated.

Definition B.4.12 (Defect Formation)**.** *Formations that are not intact are defect.*

Lemma B.4.10. *Let $F \in \mathfrak{F}_{\mathcal{V}}$ then in holds for all $\forall F' \in \mathfrak{F}_{\mathcal{V}}$ that*

$$\mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,,\bigcup F'\right) \leqslant \mathrm{cnd}_{\mathcal{V}}(F') \leqslant \mathrm{cnd}_{\mathcal{V}}(F) + \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,,\bigcup F'\right) \tag{184}$$

TODO: Proof upper bounds.

Proof. For $\mathrm{cnd}_{\mathcal{V}}(F') \neq 0$ and $\mathrm{cnd}_{\mathcal{V}}(F') \neq 0$ it holds that

$$\begin{aligned}
& \mathrm{cnd}_{\mathcal{V}}(F') = \max\left(\ \left\{\, v \in \mathbb{N} \,\middle|\, \#\mathrm{clrd}_v^{\mu_{\mathcal{V}}}(F') \geqslant 1 \right\} \cup \{0\}\ \right) \\
\Rightarrow\ & \exists I' \in F' : \mathrm{cnd}_{\mathcal{V}}(F') = \mu_{\mathcal{V}}(I')
\end{aligned}$$

and analog

$$\exists I \in F : \mathrm{cnd}_{\mathcal{V}}(F) = \mu_{\mathcal{V}}(I)$$

so that

$$\begin{aligned}
&\mu_{\mathcal{V}}(I') \geqslant \mu_{\mathcal{V}}(I) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(I, I'\right) \\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F') \geqslant \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(I, I'\right) \\
\Rightarrow\ &\mathrm{err}^{\mu_{\mathcal{V}}}\left(I, I'\right) \geqslant \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{cnd}_{\mathcal{V}}(F') \\
\Rightarrow\ &\mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) \geqslant \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{cnd}_{\mathcal{V}}(F') \\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) \leqslant \mathrm{cnd}_{\mathcal{V}}(F')
\end{aligned}$$

For the remaining cases the following holds: If $\mathrm{cnd}_{\mathcal{V}}(F) = 0$ the lower bound is trivially true since negative. If $\mathrm{cnd}_{\mathcal{V}}(F) \neq 0$ and $\mathrm{cnd}_{\mathcal{V}}(F') = 0$ then either $\forall I' \in F' : \mu_{\mathcal{V}}(I') = 0$ and thus $\mu_{\mathcal{V}}(\bigcup F') = 0$ due to σ-additivity or $F' = \varnothing$ and thus

$$\mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) = \mu_{\mathcal{V}}(\bigcup F) \geqslant \mu_{\mathcal{V}}(I) = \mathrm{cnd}_{\mathcal{V}}(F)$$

so that the lower bound is non-positive and thus also trivially true. Proving the upper bound is similar using the same case decomposition as above, first

$$\begin{aligned}
&\mu_{\mathcal{V}}(I') \leqslant \mu_{\mathcal{V}}(I) + \mathrm{err}^{\mu_{\mathcal{V}}}\left(I, I'\right) \\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F') \leqslant \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(I, I'\right) \\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{cnd}_{\mathcal{V}}(F') \leqslant \mathrm{err}^{\mu_{\mathcal{V}}}\left(I, I'\right) \\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{cnd}_{\mathcal{V}}(F') \leqslant \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) \\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F) + \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) \geqslant \mathrm{cnd}_{\mathcal{V}}(F')
\end{aligned}$$

further if $\mathrm{cnd}_{\mathcal{V}}(F') = 0$ the upper bound is trivially true, further $\mathrm{cnd}_{\mathcal{V}}(F') \neq 0$ and $\mathrm{cnd}_{\mathcal{V}}(F) = 0$

$$\mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) = \mu_{\mathcal{V}}(\bigcup F') \geqslant \mu_{\mathcal{V}}(I') = \mathrm{cnd}_{\mathcal{V}}(F')$$

so that the upper bound is also true.

□

Lemma B.4.11. *Let $F \in \mathfrak{F}_{\mathcal{V}}$ then in holds for all $\forall F' \in \mathfrak{F}_{\mathcal{V}}$ that*

$$\mathrm{crr}_{\mathcal{V}}(F) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) \leqslant \mathrm{crr}_{\mathcal{V}}(F') \leqslant \mathrm{crr}_{\mathcal{V}}(F) + \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,, \bigcup F'\right) \tag{185}$$

Proof. The proof is analog to the of lemma B.4.10 since the definitions of condition and corruption differ in changing a $\geqslant$ to a $>$ which is actual not relevant for the proof above, since only the existence of the required significations is used. □

Lemma B.4.12. *Let $F \in \mathfrak{F}_{\mathcal{V}}$ then in holds for all $\forall F' \in \mathfrak{F}_{\mathcal{V}}$ that*

$$\mathrm{fi}_{\mathcal{V}}(F) - \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,,\bigcup F'\right) \leqslant \mathrm{fi}_{\mathcal{V}}(F') \leqslant \mathrm{fi}_{\mathcal{V}}(F) + \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,,\bigcup F'\right) \tag{186}$$

Proof. Let $l := \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,,\bigcup F'\right)$ then

$$\begin{aligned}
\mathrm{fi}_{\mathcal{V}}(F) - l &= \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F)\,,\,\mathrm{cnd}_{\mathcal{V}}(F) + 1 - \mathrm{bft}_{\mathcal{V}}\,) - l \\
&= \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F) - l\,,\,\mathrm{cnd}_{\mathcal{V}}(F) - l + 1 - \mathrm{bft}_{\mathcal{V}}\,) \\
&\leqslant \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F')\,,\,\mathrm{cnd}_{\mathcal{V}}(F) - l + 1 - \mathrm{bft}_{\mathcal{V}}\,) && \text{(By lemma B.4.11)} \\
&\leqslant \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F')\,,\,\mathrm{cnd}_{\mathcal{V}}(F') + 1 - \mathrm{bft}_{\mathcal{V}}\,) && \text{(By lemma B.4.10)} \\
&= \mathrm{fi}_{\mathcal{V}}(F')
\end{aligned}$$

and for the upper bound

$$\begin{aligned}
\mathrm{fi}_{\mathcal{V}}(F) + l &= \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F)\,,\,\mathrm{cnd}_{\mathcal{V}}(F) + 1 - \mathrm{bft}_{\mathcal{V}}\,) + l \\
&= \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F) + l\,,\,\mathrm{cnd}_{\mathcal{V}}(F) + l + 1 - \mathrm{bft}_{\mathcal{V}}\,) \\
&\geqslant \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F')\,,\,\mathrm{cnd}_{\mathcal{V}}(F) + l + 1 - \mathrm{bft}_{\mathcal{V}}\,) && \text{(By lemma B.4.11)} \\
&\geqslant \min(\,\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F')\,,\,\mathrm{cnd}_{\mathcal{V}}(F') + 1 - \mathrm{bft}_{\mathcal{V}}\,) && \text{(By lemma B.4.10)} \\
&= \mathrm{fi}_{\mathcal{V}}(F')
\end{aligned}$$

□

B.5. Semiotics

This section[45] contains a discussion on the interoperability of a formation. First we discuss the two classes of significations, the denoting[46] and the connoting significations.

B.5.1. Denoting Significations

Definition B.5.1 (Set of Denoting Significations)**.** *Let $\mathcal{V}$ be a sourced key value structure and $F \in \mathfrak{F}_{\mathcal{V}}$ then its set of denoting significations is defined as*

$$\mathrm{ds}_{\mathcal{V}}(F) := \begin{cases} \mathrm{cf}_{\mathcal{V}}(F) & \text{if } \#\mathrm{fv}_{\mathcal{V}}(\mathrm{cf}_{\mathcal{V}}(F)) = 1 \\ \varnothing & \text{otherwise} \end{cases} \tag{187}$$

B.5.2. Connoting Significations

Definition B.5.2 (Set of Connoting Significations)**.** *Let $\mathcal{V}$ be a sourced key value structure and $F \in \mathfrak{F}_{\mathcal{V}}$ then its set of connoting significations is defined as*

$$\mathrm{cs}_{\mathcal{V}}(F) := F \backslash \mathrm{ds}_{\mathcal{V}}(F) \tag{188}$$

[45] It is called semiotics, since it provides the means of semiotic decoding. However, the actual denotations as references to real world objects are not discussed, since, even by name the sourced key value structure is a concept of pure information processing, that can refer to objects of the real world. So soundness as a correspondences to the objects of the real world is not discussed.

[46] Here the words denotation and connotation are intentionally avoided. In semiotics they refer to real world objects or types thereof, but this is theory is about pure information processing only, and not how it is mapped to the world.

B.5.3. Interpretation

Definition B.5.3 (Interpretation[47])**.** *Let $\mathcal{V}$ be a sourced key value structure and $F \in \mathfrak{F}_\mathcal{V}$ then*

$$\text{inter}_\mathcal{V}(F) := \text{fv}_\mathcal{V}(\text{ds}_\mathcal{V}(F)) \tag{189}$$

is the interpretation of F.

Corollary B.5.1. *The interpretation of a formation is either empty or contains a single value.*

Corollary B.5.2. *Only definite formations have non empty interpretations.*

Corollary B.5.3. *Gibberish[48] has no interpretation.*

Corollary B.5.4. *Using a connoting signification for interpretation is a fallacy.*

The above corollary might seem like a fallacy of spurious accuracy. Obviously, in the real world it is sometimes correct to use the connoting signification for interpretation, that is not exactly what the corollary states. The mentioned ostensible real world counter examples usually involve coded language to ensure that only persons with sufficient context knowledge should be able to reconstruct the actual meaning and thus use different formations for interpretation others. This corollary therefore must applied relative to a given context that selects formations. Of course there could be further language plays that try to maximise the amounts of formations under which something is interpretable. On the other hand the intention of this theory is to analyses of pure information processing as needed network protocol analysis.

B.5.4. The Fundamental Significations Decomposition

The previous definitions yield the following decomposition corollary:

Corollary B.5.5 (The Fundamental Significantions Decomposition)**.**

$$\forall F \in \mathfrak{F}_\mathcal{V} : \text{ds}_\mathcal{V}(F) \cup \text{cs}_\mathcal{V}(F) = F \wedge \text{ds}_\mathcal{V}(F) \cap \text{cs}_\mathcal{V}(F) = \varnothing \tag{190}$$

B.5.5. Significance Level Analysis

This section provides some formal tools for denoting significations and connoting significations calculation that are required later. It is based on the following definition:

Definition B.5.4 (Significance Level Sets)**.** *Let $\mathcal{V}$ be a sourced key value structure, and $s \in \mathbb{N}$ denote a signification level, then the signification level s set is defined as*

$$\begin{aligned} &\text{sls}^s_\mathcal{V} : \mathfrak{F}_\mathcal{V} \to \mathfrak{F}_\mathcal{V} \\ &\text{sls}^s_\mathcal{V} := F \mapsto \{S \in F \,|\, \mu_\mathcal{V}(S) = s\} \end{aligned} \tag{191}$$

[47] That is not not necessarily the interpretation of a source, but of the interpreter, that might be a source, of a formation.

[48] Where a formation is gibberish, if it is not empty and only consists of significations that have zero significance, i.e. only non definite associations.

Corollary B.5.6. *The signification level sets are well defined.*

Proof. This proof is analog to the on of lemma $B.4.1$. □

Corollary B.5.7 (Alternative Significance Level Sets Definition)**.** *Let $\mathcal{V}$ be a sourced key value structure,*

$$\forall s \in \mathbb{N} : \mathrm{sls}^s_{\mathcal{V}}(F) = \mathrm{clrd}^{\mu\nu}_s(F) \backslash \mathrm{clrd}^{\mu\nu}_{s+1}(F) \tag{192}$$

Corollary B.5.8. *Let $\mathcal{V}$ be a sourced key value structure, then it holds that*

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : \forall v, v' \in \mathbb{N} : v \neq v' \Rightarrow \mathrm{sls}^v(F) \cap \mathrm{sls}^{v'}(F) = \varnothing \tag{193}$$

Lemma B.5.1 (Significance Level Decomposition)**.**

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : F = \bigcup_{s=0}^{\infty} \mathrm{sls}^s_{\mathcal{V}}(F) = \bigcup_{s=0}^{\mathrm{cnd}_{\mathcal{V}}(F)} \mathrm{sls}^s_{\mathcal{V}}(F) \tag{194}$$

Proof. The first equality follows from the fact that 0 is the lowest possible value of a meassure, so that for each $S \in F$ there exists, due to the finiteness of $\Omega_{\mathcal{V}}$ a $s \in \mathbb{N}$ with $s \in \mu_{\mathcal{V}}(S)$ for every $S \in F$. The second equality is a proof by contradiction. Assume there was $S' \in F$ so that

$$S' \notin \bigcup_{s=0}^{\mathrm{cnd}_{\mathcal{V}}(F)} \mathrm{sls}^s_{\mathcal{V}}(F)$$

then it follows that $\mu_{\mathcal{V}}(S') > \mathrm{cnd}_{\mathcal{V}}(F)$, however lemma B.4.5 yields the contradiction $\mu_{\mathcal{V}}(S') \leqslant \mathrm{cnd}_{\mathcal{V}}(F)$. □

Lemma B.5.2 (Significance Fissure)**.** *Let $\mathcal{V}$ be a sourced key value structure, then it holds that*

$$\forall F \in \mathfrak{F} : \forall s \in \mathbb{N} : \mathrm{crr}_{\mathcal{V}}(F) < s < \mathrm{cnd}_{\mathcal{V}}(F) \Rightarrow \mathrm{slf}^s_{\mathcal{V}}(F) = \varnothing \tag{195}$$

Proof. Let $F \in \mathfrak{F}$ then by lemma B.4.7 it holds that

$$\forall s \in \mathbb{N} : \mathrm{crr}_{\mathcal{V}}(F) < s < \mathrm{cnd}_{\mathcal{V}}(F) + 1 \Rightarrow \#\mathrm{clrd}^{\mu\nu}_s(F) = 1$$

and using lemma B.4.2 for a proof by contradiction[49] yields

$$\forall v \in \mathbb{N} : \mathrm{crr}_{\mathcal{V}}(F) < v < \mathrm{cnd}_{\mathcal{V}}(F) \Rightarrow \mathrm{clrd}^{\mu\nu}_v(F) = \mathrm{clrd}^{\mu\nu}_{v+1}(F)\ .$$

and this directly implies that

$$\forall v \in \mathbb{N} : \mathrm{crr}_{\mathcal{V}}(F) < v < \mathrm{cnd}_{\mathcal{V}}(F) \Rightarrow \mathrm{clrd}^{\mu\nu}_s(F) \backslash \mathrm{clrd}^{\mu\nu}_{s+1}(F) = \varnothing$$

and thus by B.5.7 implies this lemma. □

[49] Assume that $\mathrm{clrd}^{\mu\nu}_s(F) \supset \mathrm{clrd}^{\mu\nu}_{s+1}(F)$, then there exists $I \in \mathrm{clrd}^{\mu\nu}_s(F) \backslash \mathrm{clrd}^{\mu\nu}_{s+1}(F)$ which contradicts $\#\mathrm{clrd}^{\mu\nu}_s(F) = 1$, since otherwise there would exist a further $I' \in \mathrm{clrd}^{\mu\nu}_s(F)$ with $I \neq I'$ due to $\#\mathrm{clrd}^{\mu\nu}_{s+1}(F) = 1$.

Lemma B.5.3 (Condition Level Separation)**.**

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : F = \mathrm{sls}_{\mathcal{V}}^{\mathrm{cnd}_{\mathcal{V}}(F)}(F) \cup \bigcup_{s=0}^{\mathrm{crr}_{\mathcal{V}}(F)} \mathrm{sls}_{\mathcal{V}}^{s}(F) \tag{196}$$

Proof. Using B.5.1 yields:

$$\begin{aligned} F &= \bigcup_{s=0}^{\mathrm{cnd}_{\mathcal{V}}(F)} \mathrm{sls}_{\mathcal{V}}^{s}(F) \\ &= \mathrm{sls}_{\mathcal{V}}^{\mathrm{cnd}_{\mathcal{V}}(F)}(F) \cup \bigcup_{s=0}^{\mathrm{cnd}_{\mathcal{V}}(F)-1} \mathrm{sls}_{\mathcal{V}}^{s}(F) \\ &= \mathrm{sls}_{\mathcal{V}}^{\mathrm{cnd}_{\mathcal{V}}(F)}(F) \cup \bigcup_{s=\mathrm{cnd}_{\mathcal{V}}(F)+1}^{\mathrm{cnd}_{\mathcal{V}}(F)-1} \mathrm{sls}_{\mathcal{V}}^{s}(F) \cup \bigcup_{s=0}^{\mathrm{crr}_{\mathcal{V}}(F)} \mathrm{sls}_{\mathcal{V}}^{s}(F) \\ &= \mathrm{sls}_{\mathcal{V}}^{\mathrm{cnd}_{\mathcal{V}}(F)}(F) \cup \bigcup_{s=0}^{\mathrm{crr}_{\mathcal{V}}(F)} \mathrm{sls}_{\mathcal{V}}^{s}(F) \qquad \text{(By lemma B.5.2)} \end{aligned}$$

□

Corollary B.5.9 (Alternative Cleared Formation Definition)**.**

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : \mathrm{cf}_{\mathcal{V}}(F) = \mathrm{sls}_{\mathcal{V}}^{\mathrm{cnd}_{\mathcal{V}}(F)}(F) \cup \bigcup_{s=\mathrm{bft}_{V}}^{\mathrm{crr}_{\mathcal{V}}(F)} \mathrm{sls}_{\mathcal{V}}^{s}(F) \tag{197}$$

Corollary B.5.10.

$$\forall F \in \mathfrak{F}_{\mathcal{V}} : \mathrm{crr}_{\mathcal{V}}(F) < \mathrm{bft}_{\mathcal{V}} \Rightarrow \mathrm{cf}_{\mathcal{V}}(F) = \mathrm{sls}_{\mathcal{V}}^{\mathrm{cnd}_{\mathcal{V}}(F)}(F) \tag{198}$$

Lemma B.5.4 (Denoting significations From Definitines)**.**

$$\#\mathrm{cf}_{\mathcal{V}}(F) = 1 \Rightarrow \mathrm{ds}_{\mathcal{V}}(F) = \mathrm{cf}_{\mathcal{V}}(F) \tag{199}$$

Proof. To to the premise the following is well defined

$$\{S\} := \mathrm{cf}_{\mathcal{V}}(F)$$

and directly implies

$$\mu_{\mathcal{V}}(S) \geqslant \mathrm{bft}_{\mathcal{V}} \geqslant 1$$

so that by lemma B.3.3 it follows that

$$\#\mathrm{sv}_{\mathcal{V}}(S) = 1$$

so that

$$\#\mathrm{fv}_{\mathcal{V}}(\mathrm{cf}_{\mathcal{V}}(F)) = \#\mathrm{fv}_{\mathcal{V}}(\{S\}) = \#\mathrm{sv}_{\mathcal{V}}(S) = 1 \ .$$

This reduces the definition of $\mathrm{ds}_{\mathcal{V}}(F)$ to one case, which can be calculated as:

$$\mathrm{ds}_{\mathcal{V}}(F) = \mathrm{cf}_{\mathcal{V}}(F)$$

□

B.6. Interpretation Resilience

Resilience in this context is the ability of interpretation to remain unchanged with respect to modifications of used associations. resilience

Theorem B.6.1 (Interpretation Resilience Theorem)**.** *Let $\mathcal{V}$ be a sourced key value structure and $F \in \mathfrak{F}_{\mathcal{V}}$ then it holds that*

$$\forall F' \in \mathfrak{F}_{\mathcal{V}} : \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F\,,\bigcup F'\right) < \mathrm{fi}_{\mathcal{V}}(F) \Rightarrow \mathrm{inter}_{\mathcal{V}}(F) = \mathrm{inter}_{\mathcal{V}}(F') \tag{200}$$

where $\bigcup F \in \mathfrak{A}$ yields the set of all associations the formation F is build from. The above error is thus a measure of the miss-match in associations between formations.

Proof. The proof is longer, but has the following staged outline structure:

1. Prove that

$$\exists r^* \in \mathrm{codom}(\mathrm{sr}_{\mathcal{V}}) : \mathrm{sr}_{\mathcal{V}}(\mathrm{ds}_{\mathcal{V}}(F)) = \{\{r^*\}\} \wedge \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\} \subseteq \mathrm{cf}_{\mathcal{V}}(F')\ . \tag{201}$$

2. Use eqn. 201 that prove that

$$\forall r \in \mathrm{sr}_{\mathcal{V}}(F) \backslash \{\{r^*\}\} : [r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \notin \mathrm{cf}_{\mathcal{V}}(F') \tag{202}$$

3. Use eqn. 202 that prove that

$$\forall r \in \mathrm{codom}(\mathrm{sr}_{\mathcal{V}}) \backslash \{r^*\} : [r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \notin \mathrm{cf}_{\mathcal{V}}(F') \tag{203}$$

4. Finally, use eqn. 203 prove that

$$\mathrm{inter}_{\mathcal{V}}(F) = \mathrm{inter}_{\mathcal{V}}(F')\ .$$

For each of the following proof stages is required to keep that, since F is definite due to corollary B.4.6, there exists a signification relation $r^* \in \mathrm{img}(\mathrm{sr}_{\mathcal{V}})$ so that

$$\{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}\} := \mathrm{cf}_{\mathcal{V}}(F) \tag{204}$$

is well defined and thus lemma B.5.4 together with corollary B.5.10 and definition B.5.4 implies that

$$\mathrm{ds}_{\mathcal{V}}(F) = \mathrm{cf}_{\mathcal{V}}(F) = \mathrm{sls}^{\mathrm{cnd}_{\mathcal{V}}(F)}_{\mathcal{V}}(F) = \{I \in F | \mu_{\mathcal{V}}(I) = \mathrm{cnd}_{\mathcal{V}}(F)\}\ . \tag{205}$$

The proofs of the stages in particular are

1. Equation 204 and using nested point wise application implies that,

$$\mathrm{sr}_{\mathcal{V}}(\mathrm{ds}_{\mathcal{V}}(F)) = \{\{r^*\}\}$$

which is the first part of what is to be showed for this stage and further

$$\mu_{\mathcal{V}}\left([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}\right) = \mathrm{cnd}_{\mathcal{V}}(F)\ . \tag{206}$$

and thus it holds

$$\begin{aligned}
&\mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \geqslant \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mathrm{err}^{\mu_{\mathcal{V}}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}, [r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Lemma A.4.2)}\\
\Rightarrow\ &\mathrm{err}^{\mu_{\mathcal{V}}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}, [r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \geqslant \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'})\\
\Rightarrow\ &\mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F, \bigcup F'\right) \geqslant \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Corollary A.5.1)}\\
\Rightarrow\ &\mathrm{fi}^{\mu_{\mathcal{V}}}(F) > \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Premise of 200)}\\
\Rightarrow\ &\min(\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F), \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{bft}_{\mathcal{V}} + 1) > \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'})\\
& && \text{(Definition B.4.10)}\\
\Rightarrow\ &\mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{bft}_{\mathcal{V}} + 1 > \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'})\\
\Rightarrow\ &\mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) > \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mathrm{cnd}_{\mathcal{V}}(F) + \mathrm{bft}_{\mathcal{V}} - 1\\
\Rightarrow\ &\mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) > \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{cnd}_{\mathcal{V}}(F) + \mathrm{bft}_{\mathcal{V}} - 1 && \text{(By 206)}\\
\Rightarrow\ &\mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) > \mathrm{bft}_{\mathcal{V}} - 1\\
\Rightarrow\ &\mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \geqslant \mathrm{bft}_{\mathcal{V}}
\end{aligned}$$

so that $\mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \geqslant \mathrm{bft}_{\mathcal{V}}$ which implies that $[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \in \mathrm{cf}_{\mathcal{V}}(F')$. Since there is only a value and thus key in $\mathrm{ds}_{\mathcal{V}}(F)$ this proofs the target equation 201 of this stage.

2. Equation 204 together with lemma B.5.3 implies that that for all $[r]^{\mathrm{sr}_{\mathcal{V}}}_{F} \in F$ with $r \neq r^*$ that

$$\mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}) \leqslant \mathrm{crr}_{\mathcal{V}}(F)\ , \tag{207}$$

so that

$$\begin{aligned}
&\mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) \geqslant \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mathrm{err}^{\mu_{\mathcal{V}}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}, [r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Lemma A.4.2)}\\
\Rightarrow\ &\mathrm{err}^{\mu_{\mathcal{V}}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}, [r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \geqslant \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F})\\
\Rightarrow\ &\mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F, \bigcup F'\right) \geqslant \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Corollary A.5.1)}\\
\Rightarrow\ &\mathrm{fi}^{\mu_{\mathcal{V}}}(F) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) && \text{(Premise of 200)}\\
\Rightarrow\ &\min(\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F), \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{bft}_{\mathcal{V}}) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F} + 1)\\
& && \text{(Definition B.4.10)}\\
\Rightarrow\ &\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F) > \mu_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F})\\
\Rightarrow\ &\mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mathrm{bft}_{\mathcal{V}} + \mathrm{crr}_{\mathcal{V}}(F)\\
\Rightarrow\ &\mathrm{crr}_{\mathcal{V}}(F) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mathrm{bft}_{\mathcal{V}} + \mathrm{crr}_{\mathcal{V}}(F) && \text{(By 207)}\\
\Rightarrow\ &\mathrm{bft}_{\mathcal{V}} > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'})\ .
\end{aligned}$$

which implies that $\mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \notin \mathrm{cf}_{\mathcal{V}}(F')$ for all $r \neq r^*$ with $[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F} \in F$, and thus this proof this stages target equation 202.

3. For all $r \in \mathrm{codom}(\mathrm{sr}_{\mathcal{V}})$ with $[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F} \notin F$ it holds that $[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F} = \varnothing$ and therefore

$$
\begin{aligned}
& \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) \geqslant \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mathrm{err}^{\mu_{\mathcal{V}}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}, [r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Lemma A.4.2)} \\
\Rightarrow\ & \mathrm{err}^{\mu_{\mathcal{V}}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}, [r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \geqslant \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) \\
\Rightarrow\ & \mathrm{err}^{\mu_{\mathcal{V}}}\left(\bigcup F, \bigcup F'\right) \geqslant \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) && \text{(Corollary A.5.1)} \\
\Rightarrow\ & \mathrm{il}^{\mu_{\mathcal{V}}}(F) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) && \text{(Premise of 200)} \\
\Rightarrow\ & \min(\mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F), \mathrm{cnd}_{\mathcal{V}}(F) - \mathrm{bft}_{\mathcal{V}} + 1) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}) \\
& && \text{(Definition B.4.10)} \\
\Rightarrow\ & \mathrm{bft}_{\mathcal{V}} - \mathrm{crr}_{\mathcal{V}}(F) > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) - \mu_{\mathcal{V}}(\varnothing) \\
\Rightarrow\ & \mathrm{bft}_{\mathcal{V}} > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) + \mathrm{crr}_{\mathcal{V}}(F) \\
\Rightarrow\ & \mathrm{bft}_{\mathcal{V}} > \mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'})
\end{aligned}
$$

which implies that $\mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \notin \mathrm{cf}_{\mathcal{V}}(F')$ so that $\mu_{\mathcal{V}}([r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \notin \mathrm{cf}_{\mathcal{V}}(F')$. Merging these cases of $r \in \mathrm{codom}(\mathrm{sr}_{\mathcal{V}})$ with the ones of target equation 202 yields the target equation 203 of this stage.

4. The last stage is started by expressing F' though its signification relations[50] in the following way

$$
\begin{aligned}
F' &= \{[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \mid \{\{r\}\} \in \mathrm{sr}_{\mathcal{V}}(F')\} \\
&= \{[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \mid \{\{r\}\} \in \mathrm{sr}_{\mathcal{V}}(F') \wedge (r \neq r^* \vee r = r^*)\} \\
&= \{[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \mid (\{\{r\}\} \in \mathrm{sr}_{\mathcal{V}}(F') \wedge r \neq r^*) \vee (r \in \mathrm{sr}_{\mathcal{V}}(F') \wedge r = r^*)\} \\
&= \{[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \mid (\{\{r\}\} \in \mathrm{sr}_{\mathcal{V}}(F') \wedge r \neq r^*)\} \cup \{[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \mid (\{\{r\}\} \in \mathrm{sr}_{\mathcal{V}}(F') \wedge r = r^*)\} \\
&= F' \backslash \mathrm{cf}_{\mathcal{V}}(F') \cup \{[r]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'} \mid \{\{r\}\} \in \mathrm{sr}_{\mathcal{V}}(F') \wedge r = r^*\} && \text{(By Eqn. 203)} \\
&= F' \backslash \mathrm{cf}_{\mathcal{V}}(F') \cup \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\}
\end{aligned}
$$

and therefore

$$
\begin{aligned}
\mathrm{cf}_{\mathcal{V}}(F') &= F' \cap \mathrm{cf}_{\mathcal{V}}(F') \\
&= (F' \backslash \mathrm{cf}_{\mathcal{V}}(F') \cup \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\}) \cap \mathrm{cf}_{\mathcal{V}}(F') \\
&= (F' \backslash \mathrm{cf}_{\mathcal{V}}(F') \cap \mathrm{cf}_{\mathcal{V}}(F')) \cup (\{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\} \cap \mathrm{cf}_{\mathcal{V}}(F')) \\
&= \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\} \cap \mathrm{cf}_{\mathcal{V}}(F') \\
&= \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\}\ .
\end{aligned}
$$

[50] The relations are computed by nesting the point-wise application syntax.

With this we can calculate $\mathrm{cf}_{\mathcal{V}}(F')$ and $\mathrm{cf}_{\mathcal{V}}(F)$. For $\mathrm{cf}_{\mathcal{V}}(F')$ this is

$$\begin{aligned}\mathrm{fv}_{\mathcal{V}}(\mathrm{cf}_{\mathcal{V}}(F')) &= \mathrm{fv}_{\mathcal{V}}(\{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\}) \\ &= \bigcup_{I \in \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\}} \mathrm{sv}_{\mathcal{V}}(I) \\ &= \mathrm{sv}_{\mathcal{V}}([r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}) \\ &= \{r_1^*\}\end{aligned}$$

and analog

$$\mathrm{fv}_{\mathcal{V}}(\mathrm{cf}_{\mathcal{V}}(F)) = \{r_1^*\} \ .$$

This allows to direcly compute the denoting significations of F and F' using the denoting signification definition, which yields

$$\begin{aligned}\mathrm{ds}_{\mathcal{V}}(F) = \mathrm{cf}_{\mathcal{V}}(F) = \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F}\} \\ \mathrm{ds}_{\mathcal{V}}(F') = \mathrm{cf}_{\mathcal{V}}(F') = \{[r^*]^{\mathrm{sr}_{\mathcal{V}}}_{\bigcup F'}\}\end{aligned}$$

so that applying the interpretation definition yields

$$\begin{aligned}\mathrm{inter}_{\mathcal{V}}(F) = r_1^* \\ \mathrm{inter}_{\mathcal{V}}(F') = r_1^*\end{aligned}$$

which implies

$$\mathrm{inter}_{\mathcal{V}}(F) = \mathrm{inter}_{\mathcal{V}}(F')$$

and thus concludes this proof.

□

B.7. Byzantine Interpretation Failure Tolerance

This section provides the justification to call the significance threshold of the sourced key value structures byzantine failure threshold. A byzantine failure in this semiotic context is an event in with sources successfully conspire to intentionally change the value of a key for a interpreter, while persevering the validity of the interpreters interpretation. In this failure case the interpreter does a valid inference of a keys value, thus does not make any formal errors, but gets an unsound result anyway. The byzantine failure in this context is thus an interpretation failure that can only be discussed with reference to some truth define by reference to real world objects. While this discussion aims to keep the real world objects outside of the realm of a discussion, we just define the type of formation that a complete data set of truthfully reporting of sources using the same denotations, will provide:

Theorem B.7.1 (Byzantine Interpretation Failure Tolerance)**.** *Let $\mathcal{V}$ be a sourced key value structure and $F^* \in \mathfrak{F}_\mathcal{V}$ be a intact formation then it holds that*

$$\forall F \in \mathfrak{F}_\mathcal{V} : \mathrm{err}^{\mu_\mathcal{V}} \left(\bigcup F^* , \bigcup F \right) < \mathrm{bft}_\mathcal{V} \Rightarrow \mathrm{inter}_\mathcal{V}(F^*) = \mathrm{inter}_\mathcal{V}(F) \ . \tag{208}$$

Proof. The proof from here is actually trivial, but nothing of the definition of a theorem states that it can not also be a corollary. Inserting the statement of lemma B.4.9 into the interpretation resilience theorem B.6.1 proves this theorem. □

The error in the above theorem refers to associations. By construction a source can only modify its own associations. In an digital application, this implicit assumption build into the definitions of this theory must of course be enforced by the implementation of the application. This can be done by the proper use of digital signatures.

The inability of malicious source to manipulate other sources leads to the following corollary:

Corollary B.7.1. *If less then the byzantine failure threshold sources conspire, all their possible manipulations of a sound intact formation do not change the interpretation.*

The byzantine failure threshold of a sourced key value structure therefore describe the susceptibility of an interpreter to such forms of manipulation, that are part of what is called cultural hacking [9] but have to the knowledge of the auther never been formalized.

B.8. Implications for Security Research and Practice

The above theory discusses problem of interpreting data in quite general terms, so that it applies to any kind of information processing system that uses a form of signification process. Thus it applies to the minds of sentient beings, such as humans.

The above discussion of a byzantine failures thus entails that any such information processing system is susceptible to certain forms of attacks on the signification process. There is nothing that can be done about this beside setting the byzantine failure threshold to the maximum possible. However with respect to the semiotics of natural language that would require a asses the associations of about half of the population of the language speakers and is thus in practice not viable. The same kind of problem appears holds in many other situations. Any physically realized information processes have finite computational ressources, so that any actualized information processing that involves a signification process, can usually derive denotations, from a sampling of associations and not the complete data and is therefore attackable through manipulations of this sample. In the above data this sampling is the set of associations a formation is build of.

In the following subsections a short in-formal selection of security applications of this insight are given.

B.8.1. The Semiotic Attack

A semiotic attack an attack on any information processing system that contains a signification process. In the above formal terms, the semiotic attack manipulates the association data a formation is build of the change outcomes of the targets signification process.

In this case the target can not used any internal procedures to defend against this attack since the attack works even if all of the targets information processing subsystems operate error free.

In that regard shifting the overtone window can be seen as a tried and tested form of semiotic attack on the human decision making process.

B.8.2. The Semiotic Counter Attack

While the previous discussion of the semiotic attack might give a bleak impression regarding the security implications of this theory, the implications are not that bleak at all. A form of semiotic attack can be used to counter unwanted surveillance of any kind.

To elaborate on this: Any relevant surveillance mechanism needs to process the data gathered by it and thus gives the surveilled direct access to the association data used by the surveillers signification process. This is in the nature of the surveillance process and can not be mittigated or avoided by the surveiller. Its left to the reader as an exercise to be creative with that last statement and create applications. But regarding covered surveillance some additional remarks seem to be expedient.

The plain possibility of a semiotic attack though the surveillance system restricts the usability of the data gathered by it, if the surveillance is not to be revealed. An attacker can use the surveillance system to feed it data and force it into a response. The only defense against that is to not process the data gathered[51]. The operators of a covert surveillance system might try to conceal their use of the gathered information in random looking events, however, that makes the detection of the covered surveillance an exercise in applied statistics. Further the statistical analysis can be eased by using a method the author calls prior rigging which that be discussed in an upcoming publication. In any case data of covert surveillance can either not be used or the surveillance system can remain covert only for a limited amount of time.

Last it should be mentioned that a covert surveiller can not retaliate against someone performing a semiotic counter attack without reveling the surveillance. To add insult to injury, the semiotic counter attack against covert surveillance can not practically be outlawed. Even if the covert survellience is legal, practically outlawing the semiotic counter attack would require the legal system to reveal the surveillance in order to enforce the law.

So while covert surveillance sure has its applications, especially if the covertness in only required for a limited amount of time, as for legal prosecution or in wartimes, it is of no use to any kind of ruler as a permanent tool of power preservation.

[51] This is similar to a proposal to avoid Plain hijacking reported by Paul Watzlawick [8]. In the 1960s there where plans to implement a communication barrier between the pilots and the crew and passengers of airplanes to eliminate Plane hijacking. Forcing the pilots by blackmail into doing something is contingent on the ability of the hijackers to send information, i.e. threats. to the pilots. A sound barrier would make the usual way of performing a plain high jacking thus logically impossible.

B.9. A final Implication: Mass Surveillance is a Fools Errant.

No the above section heading is intentionally catchy and it must be clarified what exactly condones mass surveillance. Not any kind massive data collection activity is surveillance even without the mass qualification. For this section mass surveillance is an activity to gather data from everyone of a population in order to detect norm-violating behavior and take action. While taking any kind of action may not be necessarily implied by the word, with out it, mass surveillance would be pointless.

Although the semitic counter attack works on any surveillance system the foolishness of mass surveillance is not caused by the surveillance part, but by the mass part. This is clear and broadly understood consequence from statistical hypothesis testing, which is often applied in real live. Even excellent tests with an almost vanishing rate of false positives produce garbage when applied to data that contains a massive amount of samples but only a few actual positive samples. One of the most common applications of this insight is the reason why medicinal screening tests are usual not applied to the whole population but only to a selected subset of the population.

For applications as in medicine selecting the subset to perform a test on is already tricky, but at least, it can reasonably be assumed that the selection is not influenced or done by a potentially malicious adversary. In a security context this can clearly not be assumed. Further mass surveillance will allow an adversary to just "fish" for sensors to feed carefully crafted data into. Total mass surveillance makes this fishing even trivial. The adversary can reasonably assumed that anything he does is feed into the surveillance system[52].

In the following we will discuss the possible implications for internally directed and externally directed mass surveillance of state actors. Internal means surveillance of the states own population, and external means surveillance of an others states population.

B.9.1. Overt Internal Mass Surveillence

Since overt this kind of surveillance provides an overt identifiably attack interface for malicious adversaries. In the best case the adversary is just a group of punks having fun with "the system". In the worst case it is used by a external state actor to perform a high level cyberattacks on the state performing the overt surveillence. Here the fools errant turns into outright folly if the overt mass surveillance is used to justify forcing individuals into compliance with non lawfully encoded norms. A malicious adversary state can simply manipulate this system target key individuals that providing necessary services for attacked state. Depending on the extend, this makes the installation of such

[52] Beside mentioned statistical argument and the here discussed semiotic argument, there is a further system theoretic argument pointing to the same conclusion. The controllability of many systems is linked to the number of input channels and increasing the number of input channels will increase the controllability of the system in many cases. In that regard it is by no means clear why operants of surveillance systems seem to tend to assume that they are the controlling and not the controlled. They literally operate ab big potential control channel accessible by especially the surveiled adversary. But of course that discussion would need deep analysis of actual system types and is far beyond this paper.

kind of a internal mass survellince system a military grade security threat to the state implementing it. That already seems bad, but a state can still do worse, as is discussed next.

B.9.2. Covert Internal Mass Surveillence

While most prominent cases of covert internally directed mass surveillance are historical examples provided by the Gestapo and Stasi[53]. Still, it might seem that there are justifiable applications for such kinds of surveillance systems. In contrast to the above discussed overt kind, this kind will might make it harder for adversaries to detect the sensors of the surveillances systems provided, so punks messing around may not be a significant thread, but if a state power is threatened by that, the state as good as failed.

With respect to malicious extern adversaries covertness even poses a greater threat to the surveilling state that in the overt case. Covertness implies that members of the populace that are targets by the system of action can not have legal means of defensing them selfs, since that would practicality imminently end the covertness of the system. A successful malicious cyberattack of a third state of the kind mentioned in the overt case discussion, will be covered by the successfully attack surveillance system, making it an effective weapon operated and paid for by the attached. As far as security systems design failures go the author is not able to give a popper name to this kind of failure, its a kind of suicide due to fear of death construction[54]. While that may seem great for some people, especially malicious adversaries, operators of the system may feel save and in control due to their perceived power. However they should consider, that they painted a big target on them selfs, as far as the military intelligence operators of a country might be concerned. Having excellent covertness projection further just eliminated the possibility of investigating the use of force against them and what seems to be their pinnacle of power is actually their greatest weakness. Operators of such a system should not assume that they not been adequate addressed by the military in case of war, especially after the first blod of soldiers who are bound to defend their country is shed. They are not a definite threat to winning the war and good men and woman will loose their live because of them. In countries where the military is bound to a set of constitutional principles it is actually obligated to take action.

B.9.3. Overt External Mass Surveillence

For a state actor overt externally directed mass surveillance is unlikely, since a quasi declaration of war. However this case does happen for big companies, that provide services used by the masses, like for example facebook. In this case the same caveats as

[53] We Germans seem to have a knack for perseverance in really bad ideas, as it seems. That, however is just a somewhat fashionably contemporary intellectual perspective, that turns its eye away form other historical examples. For example, the gulags of the Soviet union, that in terms of terror and body-count can easily compete with, if not outright out-compete the terror systems of German creation.

[54] Really who would implement that? Most likely and idiot or a spy of the malicious adversary. If idiocy seems unlikely, there is really a shortage of well meaning reality based interpretations.

above apply, only that the effects are more economic and social without first order effects on the actual power structure of a state. But still it makes the commercial operators susceptible to sabotage by competitors.

B.9.4. Covert External Mass Surveillance.

This is basically a strategic disaster for the surveiler allowing the surveiled to use the methods mentioned for the internal directed mass surveillance section against the surveiler, where the surveiler in oder to stay covert and not starting a war has no means of or legal ground for retaliation. That is one of the games where the only winning move is not to play, if the participating actors have have sufficient nuclear armament.

B.10. Conclusion

While the results of this appendix, that somewhat accidentally fell out of a formal protocol analysis, have obvious uses for constructing many kinds of byzantine failure tolerant data processing systems, they has some deep implications as shown in the last to sections. While not yet explicitly stated, the previous sections entail a kind of morale.

Any sentient being that acts on the world can only chose between real world outcomes that are not logical contradictory. so Even gods, so they exist, are forced to make choices. In that regard wanting to have full information while persevering independent autonomy and control is logically impossible. Beings that are unable to moderate their desire for information and control will have to come to terms with the horror of realizing that the simultaneous satisfaction if their felt desires is not possible despite all the might they may posses or systems they implement.

This pattern of the creation of existential crises through a pursuit of knowledge is told in many mythological narrations of many cultures. For the West maybe the God Yok Sothoth is the most modern archetype encoding this aspect of reality. With respect to the above discussion Yok Sothoth is fittingly named the lurker at the threshold, that is that what terrifies you if the significance of the knowledge you accumulate throws you into an existential crisis.

It should be noted that it is a miss-conception that unspeakable acts need to be performed to "conjure" Yok Shothoth and gain access to the knowledge it provides. There is a simple ancient way to access this knowledge. It has its representation in the western as well in the eastern mythological and philosophical traditions. The way is to defer ones ego and accept that even the gods can only choose which of their conflicting desires they can satisfy. In the western tradition this aspiration of personal development is maybe most clearly developed in the archetype of the stoic sage, while in eastern tradition it clearly seems to be the archetype of the Buddha. The horrors and evil of Yok Sothoth are not an intrinsic aspect of the entity itself, but of those who can not defer their ego and try to sacrifice everyone and everything else just to postpone insights their ego can not handle, while still persisting that Yok Sothoth grands them knowledge - which it will.

C. Mathematics Addenda

This section contains some recapitulation and minor addenda to common mathematical concepts.

C.1. Measures

Measure space are a backbone structure of formal statistics and stochastic and further provide a nice base to formally define integrals in analysis.

C.1.1. σ-Algebras

Measures are defined on a structure called a σ-Algebra, which is motivate by captururing the properties that subvolumes represented as points of sets should have in relation to each other and the containing space:

Definition C.1.1 (σ-Algebras)**.** *Let Ω then $\mathfrak{O} \subseteq 2^\Omega$ is a σ-Algebra over Ω by definition if*

1. $\Omega \in \mathfrak{O}$
2. *If $X \in \mathfrak{O}$ then $\Omega \backslash X \in \mathfrak{O}$, that is $\mathfrak{O}$ is closed under complements.*
3. *If $\{X_i\}_{i \in \underline{N}}$ is a family with $N \in \mathbb{N}$ then $\left(\bigcap_{i \in \underline{N}} X_i\right) \in \mathfrak{O}$*

A motivation of this definition is the Banach-Tarski paradox that exemplifies that arbitrary sets of points do not yield an intuitively consistent model of entities that have volume[55]. The above is thus σ-Algrbra is thus an abstraction of the set of sets that can have the structure of a volume. With regard to this the properties mean the following:

1. Ω has a volume.
2. If X has a volume then $\Omega \backslash X$ also has one.
3. If $\{X_i\}_{i \in \underline{N}}$ is a countable family of sets that have volume then their finite intersection has a volume.

Of course the σ-Algebra is a pure structure encoding volume-likeness and has been found to be useful for not only defining volumes but measures of many other kinds, probably most notable of probability.

[55] So that the representation of Space in physics as locally a Cartesian product of real numbers, is already a model and certainly not some kind of a-priory truth. The author has the slight suspicion that the inability to merge general relativity and quantum mechanics is due to a problem at this deep level of modeling. A quantum theory of space would make the space-time manifold of relativity at least locally be the result of some random variable. If true fixing that would required finding an adequate mathematical structure for that first. In the end this was one of the major reasons for the author to turn away form great unified theories during his graduate studies and to become an applied mathematician

C.1.2. Definition

Actual measures are part of a structure called measure space that analog to the σ-Algebra defines a strcture of measures that are volume-like:

Definition C.1.2 (Measure)**.** *Let Ω be a set and $\mathfrak{O}$ a σ-Algebra over ω then μ is a measure on $\mathfrak{O}$ by definition if*

1. *$\forall X \in \mathfrak{O} : \mu(X) \geqslant 0$, that is μ is non-negative.*
2. *$\mu(\varnothing) = 0$ and*
3. *μ is countable additive, that is for all families $\{X_i\}_{i\in\mathbb{N}}$ of pairwise disjunct sets it holds that*

$$\mu\left(\bigcup_{i\in\mathbb{N}} X_i\right) = \sum_{i\in\mathbb{N}} \mu(X_i) \tag{209}$$

The above properties are again motivated by measuring volumes, in that case the encode the following semantics

1. Every volume is non-negative.
2. The empty set has volume 0.
3. Any volume of any set with volume composed of discunct sets with sub-volume is given by the addition of these sub-volumes.

Again the above structure of a measure abstract and useful for much more then volumes, i.e. for all volume-like measures, as for example most notably probability. To succinctly refer to all data required to use the concept of a measure spaces is defined as what would today best be called an interface:

Definition C.1.3 (Measure Space)**.** *A triple $(\Omega, \mathfrak{O}, \mu)$ is a measure space if $\mathfrak{O}$ is a σ-Algebra over Ω and $\mu :\to \mathbb{R} \cup \{\infty\}$ is a measure on $\mathfrak{O}$.*

C.1.3. The Principal Counting Measure

Using the powerset as σ-Algebra any countable set can be endowed with a natural measure simply counting the elements of a subset. This measure is usually called the counting measure, however during the work on this document, it has been found to be insufficiently abstract, and is abstracted a bit further, so that "the counting measure" is here is called the principal counting measure and is a counting measure:

Definition C.1.4 (Principal Counting Measure)**.** *Let $\mathfrak{U}$ be the class[56] of all sets then*

[56] $\mathfrak{U}$ is also called Universe and is the the collection of all possible sets, which is, depending on the exact formalized set theory used not itself a set. In the most commonly used ZFC theory $\mathfrak{U}$ can't be consistently defined, so an ZFC entailing, advanced theory must be used, of which there are some, NGB for example.

the principal counting measure is defined as[57]*:*

$$\begin{aligned} &\# : \{A \in \mathfrak{U} \mid |A| \in \mathbb{N}\} \mapsto \mathbb{N} \cup \{\infty\} \\ &\# := A \mapsto \begin{cases} |A| \text{ if } A \text{ is finite} \\ \infty \text{ otherwise} \end{cases} \end{aligned} \tag{210}$$

where $|A|$ *is the cardinality of the set* A*.*

C.1.4. Counting Measures

Definition C.1.5 (Counting Measure)**.** *A measure* $\mu : \mathfrak{O} \to \mathbb{N} \cup \{\infty\}$ *is a counting measure if*

$$\mu(A) = \#\{a \in A | \mu(\{a\}) = 1\} \tag{211}$$

Corollary C.1.1. *If* μ *is a counting measure then*

$$\mu(\{a\}) \neq 1 \Rightarrow \mu(\{a\}) = 0 \ . \tag{212}$$

Definition C.1.6 (Relevant Discernible)**.** *Given a counting measure space* $(\Omega, \mathfrak{O}, \mu)$ *the set of relevant discernibles for measure* μ *is:*

$$\{\, a \in \Omega \,|\, \mu(\{a\}) = 1 \,\} \subseteq \Omega \tag{213}$$

Corollary C.1.2. *Sets of Irrelevant discernibles have measure* 0*.*

Lemma C.1.1 (Counting Measure Construction)**.** *Let* Ω *be a set and* $R \subseteq \Omega$ *be a set then*

$$\mu := A \mapsto \#(A \cap R) \tag{214}$$

is a counting measure and R *is its set of relevant discernibles.*

Proof. It is a meassure due to a proof analog as given for the principal counting outcome measure and that R is the set of its relevant discernibles is trivial. □

Corollary C.1.3. *A counting meassure is the principal counting measure if and only if its set of irrelevant discernibles is empty.*

[57] Its tempting and maybe possible to generalizes # to map from the universe into the cardinal numbers as subset of the surreal numbers. But "there be dragons" of the rather hostile kind as far as advanced set theory and meta-mathematics is known to the author, who tried to construct a surreal Lebesgue measure this way once and failed. If you know of any rigor constructions of that kind, please let the Author know. Further, *NGB* is probably not sufficient for that. If functions are collections of tuples the principal counting measure then does not easily "fit" into a function. A last warning. Examination of the ideas in this footnote in detail might permanently increase your "accumulated insanity", but results in this "pure" mathematics realm would be very useful for applied mathematics.

C.2. Metrics

A metric is a kind of minimal algebraic specification on what kind of abstract interface any notion of distance ought provide, its stated here for reference to the proof that the error is a metric.

Definition C.2.1 (Metric)**.** *Let M be a set, then a mapping $d : M \times M \to \mathbb{R}$ is a metric on M by definition if for all $x, y, z \in M$ it holds that*

1. *$d(x, y) = 0 \Leftrightarrow x = y$, that is identities are indiscernible.*
2. *$d(x, y) = d(y, x)$, that is the metric is a symmetric relation.*
3. *$d(x, y) + d(y, z) \geqslant d(x, z)$, that is that the triangle inequality holds.*

The metric is provided through the following interface:

Definition C.2.2 (Metric Space)**.** *A pair (M, d) is called a metric space if d is a metric on M.*

References

[1] Ross J. Anderson, *Security Engineering: A guide to Building Dependable Distributed Systems, second edition*, Wiley Publishing Inc, 2008

[2] Information Sciences Institute University of Southern California, *RFC 793: Transmission control protocol DARPA Internet program protocol specification*, Internet Request for Comments, September 1981

[3] J. Postel, *RFC 768: User Datagram Protocol*, Internet Request for Comments, ISI, 28 August 1980

[4] S. Deering, R. Hinden, *RFC 2460: Internet Protocol, Version 6 (IPv6) Specification*, Internet Requests for Comments, Network Working Group, December 1998

[5] Information Sciences Institute University of Southern California, *RFC 791: Internet protocol DARPA Internet program protocol specification*, Internet Request for Comments, September 1981

[6] Donald E. Knuth, *The Art of Computer Programming, Volume 3 Sorting and Searching, Second Edition*, Addison-Wesley, 1998

[7] The lost sons and daughters of Nicolas Bourbaki, *Lexikon der Mathematik, Volume 2*, Spektrum Akademischer Verlag, 2001

[8] Paul Watzlawick, *Wie Wirklich is die Wirklicheit? Wahn, Täuschen, Verstehen*, 4. Auflage, Piper Verlag, 2006

[9] Thomas Düllo, Franz Liebl, *Cultural Hacking, Kunst des Strategischen Handelns*, Springer-Verlag/Wien, 2005